UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Ingeniería Técnica de Telecomunicaciones: Sonido e Imagen

PROYECTO FIN DE CARRERA

INVASIÓN ANDROIDE:
Un videojuego para Android de inmersión en la realidad.
Módulo de geolocalización.

Autor: Adrián Cereceda Díaz
Tutor: Raúl Arrabales
Director: Jorge Muñoz Fuentes

Diciembre de 2011

Índice

Capítulo 1

Introducción

Desde su invención, el teléfono móvil ha sufrido un sinfín de cambios en sus funciones, su aspecto y su hardware hasta ser lo que es hoy. En su camino ha evolucionado hasta implementar sistemas tan recientes como el WiFi, el reproductor de MP3, el GPS, la pantalla color y la pantalla táctil, llegando a convertirse en lo que actualmente conocemos como Smartphone.

Appel fue la compañía que terminó de revolucionar el mundo de los móviles con el lanzamiento de su primer iPhone, en 2007. Fue el primero en incorporar una tecnología multitáctil e hizo las delicias de los consumidores, cambiando la percepción de la calidad en los móviles, que cada vez tenían que ser piezas de tecnología más sofisticadas. Un año después, cuando se lanzó la App Store, HTC llegó con el primer teléfono en incorporar el sistema operativo Android de Google basado en Linux, diseñado originalmente para dispositivos móviles, pero que posteriormente se expandió su desarrollo para soportar otros dispositivos tales como tablets, reproductores MP3, netbooks, PC e incluso televisores.

La tecnología que se había desarrollado en los móviles permitía a los programadores crear aplicaciones y videojuegos realmente sorprendentes. Pero el mundo de los videojuegos y los móviles no siempre estuvo tan ligado como vemos hoy en día.

A finales de los años 1990 los teléfonos móviles todavía eran aparatos que solo servían para llamar. Algunos fabricantes como Nokia decidieron ofrecer algún tipo de entretenimiento en esos pequeños dispositivos que tenían botones y una pantalla LCD en blanco y negro, e introdujeron pequeños juegos basándose en las primeras arcade y consolas de principios de los 80; jamás pensaron que esto revolucionaría el mundo de los videojuegos portátiles. Estos juegos fueron evolucionando y algunos ofrecían la posibilidad de desbloquear niveles pagando al operador o conectándose a Internet. Mientras, en Japón se lanzaron los primeros móviles programables los cuales disponían de una zona de memoria donde se podían grabar datos, y se podía utilizar un lenguaje de desarrollo como Java y un cable USB o una conexión a Internet para introducir el programa en el móvil. La finalidad de esto era el desarrollo de pequeñas aplicaciones y no videojuegos. Pero aun así algunas empresas desarrollaron videojuegos en blanco y negro para esas pequeñas pantallas y resultaron un éxito. Los móviles fueron evolucionando y con ellos la memoria, potencia y los lenguajes de programación de los mismos. Actualmente el mercado de los videojuegos para móviles es más grande que cualquier otro mercado de videojuegos portátiles, teniendo cifras de ventas elevadas.

Capítulo 1

Introducción

Una de las tecnologías que definitivamente ha despegado en 2010 es la Realidad Aumentada, la cual combina elementos reales y virtuales para aumentar nuestra comprensión del mundo. Algo sacado de la ciencia ficción está en la manos de millones de usuarios en todo el mundo. No deja de ser sorprendente como un teléfono móvil, gracias a su cámara, su GPS y su pantalla se convierten en un terminal del futuro, con el que podemos apuntar a cualquier lado y obtener mágicamente más información sobre el entorno.

Teniendo en cuenta que el de los juegos es un mercado que mueve millones de dólares al año en los Estados Unidos, es comprensible que se esté apostando mucho por la realidad aumentada en este campo puesto que ésta puede aportar muchas nuevas posibilidades a la manera de jugar.

Echando la vista atrás en la evolución de los videojuegos y viendo como día a día la tecnología ofrecida se supera, parece razonable que surjan proyectos de fin de carrera como el que se explica en este documento. Un proyecto que busca ahondar en las tecnologías más punteras de los terminales móviles, a través de un videojuego que permita al usuario involucrarse de una manera más activa en este. Algo que desde tiempo atrás se viene intentando y que sólo con las tecnologías que han surgido se puede seguir trabajando.

Por tanto este proyecto irá destinado al estudio de las diferentes tecnologías que encontramos en los teléfonos móviles actualmente y cómo poder introducir estas tecnologías en un videojuego, que finalmente se creará, constituyendo la parte central de este proyecto.

1.1. Motivación del Proyecto

En este proyecto se propuso crear un videojuego para Android, que hiciera uso de las capacidades que los terminales con este sistema operativo tienen. Esto es porque, a día de hoy, no había muchos videojuegos que exprimieran al máximo las herramientas que estos dispositivos brindaban.

Android tiene un potencial bastante grande, reflejo de ello es el crecimiento experimentado en la creación de aplicaciones para Android en los últimos meses. Un potencial equiparable con el de iOs, que sí que dispone de infinidad de juegos que explotaban el potencial de las herramientas de estos teléfonos.

Es por esto que cuando se propuso abarcar este proyecto se planteó la posibilidad de crear un videojuego que aprovechara tanto los sensores de los que dispone el terminal, como del GPS que nos facilita la localización del usuario, de tal forma que se invitaba a diseñar un videojuego desde cero, dando rienda suelta a la creatividad, sin olvidar el objetivo del juego.

La idea de crear un videojuego es algo que a cualquier persona que se haya visto envuelto en el mundo de las maquinas recreativas, las enormes consolas portátiles y las videoconsolas desde pequeño, le atrae. En este caso no iba a ser diferente y desde que surgió la idea, lo primero que se hace, ya que se trataba de un videojuego destinado a terminales móviles de última generación, es investigar las posibilidades que estos te brindan y cómo puedes focalizarlas al objetivo del proyecto.

Para desarrollar el proyecto se eligió Android, ya que las prestaciones que ofrecía eran muy similares a las de los iPhones, pero no estaban siendo tan bien aprovechadas, ya que se trataba de una tecnología relativamente nueva. Además el crecimiento que estaba experimentando Android era muy elevado lo que lo hacía más atractivo si cabía. Por, otro lado el hecho de que la tecnología Smartphone con el sistema operativo Android no hubiera estallado aún en el uso de las características de estos móviles, daba el hándicap de poder hacer algo innovador para esta tecnología.

Además la implementación de un videojuego sobre la plataforma de desarrollo Android daba la comodidad de seguir aprendiendo sobre un lenguaje de programación que ya se tenía bastante dominado a lo largo de la carrera, como lo era Java. Permitiendo conocer nuevas aplicaciones y librerías de este lenguaje universal destinadas a la tecnología Smartphone.

Otro de los aspectos influyentes a la hora de desarrollar este proyecto era la posibilidad de, una vez finalizado el videojuego, comercializarlo mediante Android Market.

Capítulo 1

Introducción

Android Market es la tienda en línea de software desarrollado por Google para dispositivos Android. Un programa preinstalado en la mayoría de los dispositivos Android y que permite a los usuarios buscar y descargar aplicaciones publicadas por los desarrolladores en Android Market.

Debido a que la demanda de Smartphone con este sistema operativo está aumentando de manera impredecible, se hacía muy atractivo crear un videojuego para esta plataforma, que permitiera dar a conocer la aplicación que se desarrollara e incluso reportar con ello un beneficio económico, además del intelectual.

1.2. Objetivos

Las principales características que se deseaba explotar en este proyecto son la geo-localización mediante el uso de GPS, los sensores de los que disponen estos terminales y de herramientas que permiten introducir gráficos de alta tecnología mediante la librería desarrollada en otro proyecto sobre el que se basará éste.

Concretamente este proyecto hará uso de otro destinado a la creación de gráficos de realidad aumentada para videojuegos que implementen la capacidad de GPS. Así se cumplirá el objetivo principal de crear un videojuego que se centra en el uso de todas estas tecnologías. Un videojuego en el que el usuario interactúa con el medio con el objetivo de capturar unos robots.

Este proyecto planteaba los siguientes objetivos principales:

- Diseño de la lógica de un videojuego que permitiera el uso de las diferentes tecnologías presentes en los terminales Smartphone. De tal forma que puedan explotarse tales características de estos móviles para mejorar notablemente la experiencia del usuario.

- Uso del GPS como uno de los recursos a exprimir de estos teléfonos de última generación. Incorporándolo en el videojuego como una de las bases imprescindibles para su desarrollo.

- Uso de la librería creada en un proyecto que se desarrolla paralelamente y que permite introducir gráficos OpenGL en un videojuego de realidad aumentada insertando localizaciones geográficas, y que hace uso de los sensores de posicionamiento de un Smartphone y de la cámara integrada en el terminal.

- Uso de las librerías de Google como apoyo para la correcta consecución del juego. Haciendo uso de estas librerías se podrán añadir funcionalidades muy curiosas al videojuego que mejoraran gratamente la jugabilidad del mismo.

Al tiempo que se plantean estos objetivos surgen otros personales:

- Realización de un videojuego que permita una utilización eficiente de tecnologías de la comunicación presentes en los terminales móviles actuales como colofón de la carrera.

- Aprovechar el desarrollo del proyecto para conocer y profundizar en las funcionalidades que ofrecen nuevas herramientas de trabajo que permitan complementar los conocimientos adquiridos durante la carrera y a su vez ampliar horizontes.

Con los objetivos fijados, aparecen una serie de pasos intermedios como consecuencia de los mismos:

- Profundizar en las herramientas que se facilitan para la realización del proyecto. Se hace necesario para la creación de un videojuego conocer las posibilidades que estos terminales nos ofrecen, así como el estudio de las librerías que permiten poner en funcionamiento estas herramientas.

- Efectuar el análisis de requisitos. Se debe realizar un análisis de requisitos para el juego que determine las opciones básicas que incluirá, el comportamiento del jugador, las opciones que podrá realizar y cómo le afectarán, si habrá niveles, enemigos o ítems y cómo varía el juego en función de estos.

- Desarrollo y validación. Durante la creación del videojuego se realizarán tutorías colocadas estratégicamente para evaluar el correcto desarrollo del proyecto, perfilar los escollos que puedan surgir y buscar alternativas frente a posibles problemas insalvables. De esta manera dar una continuidad al proyecto y terminarlo en las fechas establecidas.

1.3. Contenidos de la memoria

Los contenidos de los que se compone la presente memoria están organizados en forma de capítulos. Cada uno de ellos profundiza sobre un aspecto concreto perteneciente a este proyecto. A continuación se muestra un breve resumen del contenido de cada capítulo.

Capítulo 1, Introducción. Se ofrece una primera aproximación a la idea en torno a la cual gira el proyecto, el crecimiento y la evolución de los dispositivos móviles y las nuevas posibilidades que estos terminales aportan con sus tecnologías para enriquecer la creatividad de los desarrolladores. Además, acerca al lector a los objetivos que han motivado la realización de este proyecto.

Capítulo 2, Estado de la Cuestión. Realiza un repaso a la historia de los videojuegos: empresas, juegos, consolas; ofreciendo una visión más centrada en el mundo de la telefonía móvil desde su nacimiento hasta nuestros días. También se incluye un análisis del mercado actual, en el que se realiza una comparativa con diferentes juegos y aplicaciones que pueden haber inspirado nuestro videojuego aportando ideas y clarificando el potencial de las herramientas con las que trabajamos.

Capítulo 3, Análisis de Herramientas. Incluye un análisis de la herramienta utilizada para el desarrollo del proyecto en el que se estudia a fondo el potencial de éstas para terminar de esclarecer las posibles funcionalidades que podamos implementar en nuestro proyecto.

Capítulo 4, Descripción del Videojuego. Se plantea la dinámica que seguirá el proyecto para el desarrollo del videojuego, teniendo en cuenta no sólo los requisitos objeto de este proyecto sino también los del otro proyecto que engloba la totalidad del videojuego. Por tanto se hace una descripción completa de la historia que envolverá al jugador y de las funcionalidades que tendrá disponibles.

Capítulo 5, Análisis, diseño e implementación. Recoge el análisis de la arquitectura y el modelo de conocimiento del juego incluyendo el diseño de clases, la descripción de los distintos módulos con su funcionalidad; así como el detalle de los algoritmos seleccionados para el desarrollo del juego, la aparición de ítems y sus diferentes funcionalidades y la creación de la lógica de un videojuego con todo lo que ello conlleva, niveles, puntuaciones y desarrollo.

Capítulo 6, Conclusiones. Expone las reflexiones obtenidas tras la realización de este proyecto, analizando además si se han alcanzado los objetivos marcados en un principio.

Capítulo 7, Líneas Futuras. Hace referencia a las ideas e intenciones futuras que han surgido tomando este proyecto como base; puesto que a partir de este proyecto pueden surgir nuevas ideas o incluso ampliar la funcionalidad del mismo.

Por último comentar que, al final de este documento, se encuentran los distintos Anexos que recogen la Planificación (Anexo A) y el Presupuesto (Anexo B). También un par de manuales: el de usuario (Anexo C) y el de desarrollador (Anexo D).

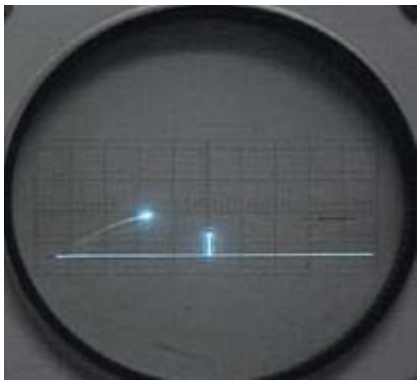
Además se incluyen dos capítulos extra en los que se adjuntan la Bibliografía y las Referencias, que presenta todas las fuentes que se han consultado para profundizar en los distintos temas expuestos en el proyecto.

Capítulo 2

Estado del Arte

2.1. Evolución de los Videojuegos

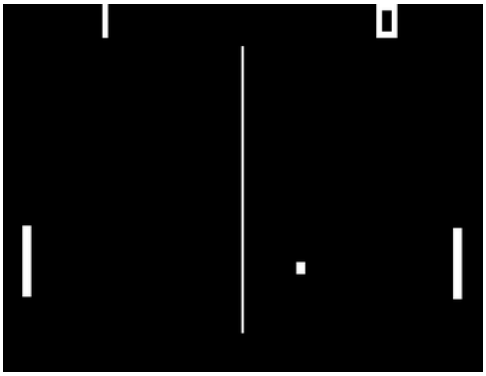
Podríamos situar el origen de los videojuegos en *"Tennis for Two"* [1], un videojuego de tenis desarrollado por William Higinbotham que usaba un osciloscopio a modo de pantalla y que fue desarrollado en 1958.



Otros textos [2] centran este origen en la genialidad que el joven estudiante Steve Rusell creó bajo el nombre "*SpaceWar*", al ser este más popular que su predecesor, a pesar de ser éste posterior, ya que se creó en 1962. Este videojuego multijugador permitía a los usuarios manejar una nave que giraba alrededor de una estrella, con el objetivo de destruir la del contrario. Muy parecido a otro juego que posteriormente fue más famoso bajo el nombre de "*Asteroids*".

Pero el nacimiento de los videojuegos se sitúa a comienzos de los setenta, época en que los grandes ordenadores, atendiendo al enorme tamaño de sus procesadores, se transformaron en dispositivos prácticamente portátiles, gracias al nacimiento del microprocesador.

En 1972 se desarrolla el primer videojuego, llamado "*PONG*" [3], que consistía en una simplificada partida de ping-pong virtual en la que se iban sumando puntos a medida que el contrincante fallaba a la hora de devolver una pelota que cruzaba de un campo a otro. Se convirtió en un juego muy popular que *Atari* comercializó en 1975 a través de la segunda consola de la historia [4].



En 1977, aprovechando el éxito que causó la *Atari Pong*, la firma Atari sacó al mercado primer sistema de videojuegos en cartucho para ordenador [5], que a pesar de que alcanzó un notable éxito comercial en EE. UU., tuvo unos inicios bastante difíciles debido a su elevado precio y al poco interés que suscitaban los videojuegos en esa época.

También empiezan a proliferar en los establecimientos de ocio (bares, salones recreativos,…) máquinas recreativas [6] que permitían jugar al "comecocos" (*Pacman*) o matar "marcianos" (*Space Invaders*), por citar algún ejemplo. Su éxito es tal, que desplazan por completo a billares o futbolines.

En las últimas décadas del siglo XX se generaliza el uso de los ordenadores personales, y con ello surgen las primeras empresas dedicadas a liderar el prometedor mercado de los videojuegos. Empresas que centran sus esfuerzos en captar nuevos clientes lo que propicia la aparición paulatina de nuevos productos cada vez de mayor calidad gráfica y sonora, con alto realismo y una gran interactividad.

Los videojuegos hoy son bastante más que un producto informático; también son un negocio, un instrumento de información y formación, un objeto de investigación y un fenómeno social.

Todo esto suscitó el mercado de la creación de las consolas, que permitieron lanzar grandes éxitos de calidad inconmensurable, como el famoso *SuperMario*, que hicieron las delicias de los niños y el éxito de la videoconsola Nintendo [7], principal empresa en este sector. Fue la primera en comercializar una consola y consiguió un gran éxito, a partir de 1985, cuando introdujo en el mercado el personaje de Mario [8]. Otras empresas como SEGA o SONY, y más tarde MICROSOFT [9], también iniciaron sus andaduras por el mundo de las consolas al tiempo que NINTENDO.

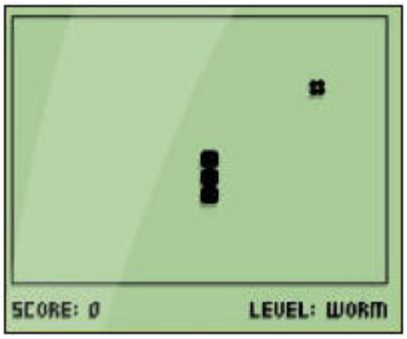
Más allá de las videoconsolas, se inició una nueva carrera por mantener la hegemonía en otro campo, el de las videoconsolas portátiles [10]. En abril de 1980 [11] aparece la primera videoconsola portátil de Nintendo, *"GAME & WATCH"*, la cual solo disponía de un juego fijo que podía visualizarse en una pequeña pantalla de LCD. Llegaron a aparecer unos 60 juegos [12] distintos durante toda la década de los 80, entre los que se encontraban títulos como *Donkey Kong y Super Mario Bros*.

Durante la década de los ochenta siguieron comercializándose estas consolas con modelos muy atractivos y originales, pero solo el lanzamiento de la GameBoy con sus cartuchos intercambiables revolucionó el mercado e hizo historia. En abril de 1989 [13] Nintendo saca al mercado japonés la GameBoy, una consola portátil, con microprocesador Z80 a 8 bits, pantalla LCD de 160x144, 4 tonos de grises, sonido, 4 pilas que le proporcionaban más de 20 horas de juego, y el Tetris incluido de serie [14]. Con este modelo y sus predecesoras, que van desde la versión de bolsillo GameBoy Pocket hasta la actual 3DS con gráficos 3D, pasando por sus diferentes versiones a color, Nintendo logró hacerse con el mercado frente a sus competidoras que, como Atari, tenían unas consolas portátiles bastante competitivas e incluso mejores gráficamente pero poco eficientes, como la Atari Lynx [15] que podía mostrar 16 colores simultáneos en su pantalla, frente a las 4 tonalidades de grises de la GameBoy, pero que hacían consumir sus seis pilas en apenas dos horas de juego. No obstante surgen otras portables que se hacen con un importante pellizco de la cuota de mercado como la PSP [16] de Sony, lanzada el día 14 de Diciembre de 2004 en Japón y 3 meses después en Norteamérica.

Las consolas portátiles constituyen un precedente sobre lo que hoy refleja el mercado de los videojuegos en los móviles. La facilidad de llevar encima una consola que te permita amenizar largas esperas o cualquier rato de tiempo muerto en cualquier lugar hacen que se conviertan en imprescindibles en el bolsillo. Pero a su vez la complejidad de muchos de los juegos y la aparatosidad de las consolas que hacen cargar con un bulto más, hacen que la evolución de este entretenimiento torne a fusionarse con los teléfonos móviles. El carácter ocasional de los videojuegos y la posibilidad de llevarlos siempre encima hacen que esto no quedara en una moda coyuntural y haya crecido día tras día.

Quizás el ejemplo más claro de fusionar las funcionalidades de los dos dispositivos lo llevara a cabo Nokia en 2003 con su N-Gage [17] que incorpora las funcionalidades como teléfono, como consola de juegos y como reproductor de mp3 y de video. Permitía jugar a múltiples jugadores mediante bluetooth o por internet e incorporaba funciones propias de una PDA.

Para llegar a este punto ha tenido que pasar mucho tiempo desde los inicios de los videojuegos en los móviles. Y es que lejos queda ya *Snake* con su serpiente echa de aquellos cuadros negros que avanzaban por la pantalla de nuestro móvil cogiendo objetos para sumar puntos. Con este juego consideramos que los videojuegos iniciaron sus andaduras en los teléfonos móviles, por lo que podremos datar el origen de los videojuegos en los teléfonos móviles en el 1997 [18]. *Snake* apareció por primera vez en el modelo 6610 de Nokia y con sus diferentes versiones [19] ha cosechado infinidad de éxitos, consiguiendo más de 400 millones de copias vendidas.



La primera etapa de los videojuegos en los móviles se vio marcada por las diferentes versiones de *Snake* y del famoso *Tetris* que se incluían en los terminales que se comercializaban entonces [20]. No obstante se volvía necesario una nueva forma de comercializarlos, que permitiera acceder a cualquier juego y descargarlo desde cualquier lugar. Por lo que se inició lo que se llaman juegos de segunda generación, los juegos WAP.

Para ello se inició el protocolo conocido como el Protocolo de Aplicación Inalámbrica (WAP) [21], un estándar de la tecnología desarrollada para permitir a los dispositivos móviles conectarse a Internet. WAP Forum fue el organismo que se encarga de desarrollar el estándar WAP, fundado por cuatro empresas del sector de las comunicaciones móviles, Sony-Ericsson, Nokia, Motorola y Openwave (originalmente Unwired Planet). Aunque el estándar tardó un tiempo en funcionar como se tenía previsto [22].

En la década de los 90 se había logrado crear un micro navegador y una versión muy reducida de la web, pero era necesario algún organismo que controlara la estandarización. Con este objetivo surgía, de la asociación de Ericsson, Nokia y Motorola, la Open Mobile Alliance [23], para garantizar la estandarización al menos en Europa, ya que en Japón, el líder en el mercado de operadores, NTT DoCoMo, se acercó con su propio estándar i-mode [24]. Ambos estándares resultaron ser muy similares, pequeños navegadores web incorporados en los terminales inalámbricos que permitían conectarse con un servidor y transferir datos al teléfono.

Evidentemente la conexión que facilitaban estos dispositivos en 1999 no llegaba ni de lejos a lo que hoy conocemos por conexión internet. La conexión que ofrecía no pasaba de cuatro líneas de texto negro sobre un fondo verde y, si había suerte, algunos gráficos monocromáticos simples. Pero en las casas mucha gente aún no se había "conectado" a la autopista de la información a la que Al Gore acaba de dar los últimos toques y que se había presentado en sociedad en 1991 [25]. Por lo tanto, la idea de navegar en el teléfono lo hacía más atractivo.

Así fue que los diseñadores de juegos comenzaron a pensar en las posibilidades que esta modalidad les brindaba y lo que podían hacer con WAP. Con esta tecnología de conexión era relativamente sencillo establecer una versión multijugador, sobre todo para juegos como el *Tres en Raya* o el *Conecta Cuatro*. Lo que resulto más complicado fue generar un intercambio de información WAP rápido para juegos como *Snake II*, aunque se logró desempeñar una versión multijugador en modo local mediante una conexión de infrarrojos [26].

Comenzaron a surgir industrias centradas en el negocio de los videojuegos para móviles, concretamente en el año 2000 nacieron dos de las grandes empresas de este campo [27]. Por un lado JAMDAT Mobile, fundada por tres ex ejecutivos de Activision, en EE.UU. Y por el otro el gran gigante de los juegos móviles que surgió de la compañía francesa Ludigames, también conocido como Ludiwap, dirigido por Michel Guillemot, y que era una empresa entre Ubisoft y Guillemot Corporation, la cual más tarde se convirtió en Gameloft [28].

Algunos títulos muy sonados en el año 2000 [29] fueron Hechicería Steve Jackson e Intercambio de peces exóticos, juegos con los que se iba trabajando hacia juegos más complejos y con un trasfondo mayor.

En el año 2001 [30] infinidad de empresas surgían para apoyar la industria de los videojuegos en los móviles. Empresas como Madden NFL, EA Sports, Electronic Arts, Hands-On Mobile o Azul Beck aportaban juegos innovadores y diseñadores con un montón de nuevas ideas.

Sin embargo, hubo un problema [31]. Existía infinidad de gente dispuesta a gastar tiempo y dinero en esta nueva modalidad de juegos, acumulando millones de minutos de datos en el proceso, pero nadie, excepto los operadores, estaba haciendo dinero.

La idea original había sido que los juegos móviles elevarían el uso de los móviles, elevando el ingreso medio por usuario, y con una parte de estos ingresos adicionales se pagaría a los desarrolladores. Desafortunadamente para los editores y desarrolladores de juegos los operadores de red no estaba de acuerdo con este enfoque.

Volviendo a los juegos, en 2001 [32] vieron la luz algunos de los mejores juegos WAP. Entre ellos *Lifestylers, El Señor de los Anillos* o *Animales de compañía inalámbricos***.**

En el verano de 2001 [33] se pudo ver lo que sería el futuro de los juegos móviles y que desbancaría al sistema WAP. En la conferencia JavaOne en San Francisco, una serie de empresas como Digital Bridges y Sega mostraron juegos en un prototipo de

Motorola en el que se ejecuta una versión micro del entorno de programación Java, conocido como Java 2 Micro Edition (J2ME [34]). J2ME demostró que los juegos de acción rápida se podían jugar en el móvil y que, a pesar de ser el futuro que sustituiría a WAP, eran tecnologías compatibles [35].

Más adelante, en 2002, se produciría uno de los acontecimientos más significativos, el lanzamiento comercial de los primeros teléfonos [36] con plataforma Java para juegos, lo que supuso su aceptación por los programadores de juegos móviles. Dos de los primeros en Europa fueron los Nokia 3410 y el Siemens M50.

Sin embargo una serie de limitaciones en la resolución de los dispositivos móviles hicieron que varias empresas siguieran brindando su apoyo a la tecnología WAP que experimentó una rápida mejora con los teléfonos de pantalla a color.

Al tiempo que la plataforma J2ME nacía lo hacían también BREW [37] en los Estados Unidos y ExEN y Mophun [38] de la mano de In-Fusio y Synergenix respectivamente. Esto era fruto del desacuerdo de estandarización, lo que ocasionaría la fragmentación en esta área.

No obstante el mercado de los juegos telefónicos seguía deleitándonos con títulos como el famoso Space Invaders, JAMDAT Bowling o Siberian Strike [39].

Si hubo un cambio crucial que se produjo en 2003, fue la adopción masiva en el mercado de teléfonos móviles con pantallas a color que parecía dar un importante paso adelante desde su introducción en 2001 [40]. Pero Nokia con el lanzamiento de N-Gage en Octubre de 2003 [41] se propuso eclipsar este avance, propuesta que se vio zafada por su elevado precio.

No obstante los teléfonos móviles estaban empezando a convertirse en piezas muy sofisticada de electrónica de consumo. En Japón, los juegos 3D se veían impulsados con títulos como [42] demostrando lo que el futuro de los juegos móviles sería. Incluso en Europa, los desarrolladores quisieron dar el salto a la tercera dimensión publicando *Extreme Air Snowboarding* [43], que fue uno de los primeros juegos pseudo-3D.

En 2004 el mercado de inversión de los juegos para móviles estaba en lo más alto, estaba inundado de dinero y se revelaron algunas de las absorciones más impresionantes [44]. Digital Chocolate compró Sumea, Mforma compró Azul Beck, Sorrent compró Macrospace y Infospace gastó unos 26 millones de dólares en América y 15 millones en Reino Unido en adquirir IOMO. Algunos como JAMDAT presentaron sus planes para una oferta pública y se pusieron a disposición en la bolsa.

Por otro lado el mercado de los videojuegos en el 2004 [45] comenzó a ser un hervidero de títulos de gran dinamismo que prometerían largas horas de entretenimiento y, con sus secuelas, sagas de mucho prestigio. Entre otros los juegos de carreras son el género más solicitado como demuestran títulos como *The Fast and the Furious, Asphalt Urban GT* y *Driv3r*.

A pesar del fracaso de Nokia con la N-Gage original, de la nada llegó Gizmondo [46], un dispositivo dedicado para juegos de móviles de características muy similares a la N-Gage. Sin embargo no era en realidad un teléfono, aunque tenía una ranura para una tarjeta SIM y sistema de apoyo para WAP, GPRS y SMS/MMS. El año 2006 comenzó con una explosión de juegos fabricados para Gizmondo que no sirvieron para nada porque la venta de tan solo 25.000 dispositivos [47] durante todo el proceso supuso una pérdida de 400 millones de dólares para sus inversores. Así terminó la trama de una consola, envuelta en una neblina de corrupción [48], que podría haber marcado precedentes introduciendo funcionalidades propias de un móvil.

Pero 2005 [49] todavía dejaba una última noticia impactante, EA se abalanzó y se compró JAMDAT por la friolera de 680 millones de dólares. Con todo esto parecía que los juegos para móviles estaban desaparecidos pero surgieron importantes títulos [50] como *Midnight Pool, Downtown Texas Hold'Em* y *Skipping Stone.*

Glu Mobile [51] completo la adquisición de móviles del Reino Unido con iFone que gracias a títulos como *Lemmings* , *V-Rally* y *Cluedo* se había convertido rápidamente en un jugador fuerte. El problema era que iFone nunca logró entrar en el mercado de los EE.UU.

Cuando llegó la revolución de juegos para móviles fue 2005 el año en que se convirtió el 3D en una propuesta del mercado de masas. Claro que los juegos 3D han estado disponibles desde 2003 [52], pero fueron necesarios un par de años para que los teléfonos tuvieran la capacidad de procesamiento suficiente como para poder disfrutarlos a pleno rendimiento.

Esto daba aún más trabajo a los programadores de juegos, ya que, además de tener que adaptarse a lanzamientos de juegos en 3D, tenían que desarrollar videojuegos en las diferentes plataformas disponibles, puesto que la portabilidad en los sistemas operativos de los terminales móviles no era tan sencilla como la experimentada con Java en los ordenadores personales. Esto era porque la diferencia de la capacidad de los

teléfonos implicaba que no todos podrían ejecutar la amplia gama de juegos de la misma manera.

Esto propicio una nueva fragmentación [53], que mientras compañías como Nokia y Motorola favorecían el estándar propuesto por Java, Sony Ericsson eligió Mascot Capsule para sus teléfonos. Pero esto no evito que los videojuegos que salieron en el 2006 [54] hicieran alarde del potencial que se estaba alcanzando, entre otros podemos encontrarnos *Tower Bloxx* o *The Fast and the Furious: Tokyo Drift*.

En muchos sentidos, las tecnologías de WAP y 3G comenzaban a estar denostadas para el mundo de los móviles y se hacía necesario un cambio que ilusionara en el mundo de los videojuegos móviles. Para vencer este escollo Nokia presento una nueva versión de su N-Gage, la N81 [55], con una versión móvil de Xbox Live, que pudo convertirse en la solución que todos esperaban. Pero a pesar de algunos títulos excelentes como *Reset Generation* , *Metal Gear Solid* y *ONE*, no se logra encender el entusiasmo de los consumidores de la forma en que Nokia espera. Y frente a los duros competidores que surgirían Nokia volvió a sufrir un nuevo fracaso [56].

La mayor sacudida para el juego móvil vino de las manos de Appel. En Junio de 2007 [57], el iPhone con pantalla táctil [58] se anunció en EE.UU. y cinco meses después en Europa, lo que aumentaba la histeria del consumo masivo. Se trataba de una hermosa pieza de la tecnología y no era ni siquiera un teléfono malo, pero lo realmente innovador para la industria de juegos móviles llegó con el lanzamiento de la App Store [59] en julio de 2008 . De repente, allí estaba una plataforma para que los consumidores pudieran comprar los juegos tan fácilmente como lo habían hecho con el MP3 a través de iTunes. También permitió a los desarrolladores vender sus juegos directamente a los consumidores sin tener que lidiar con los editores y operadores. Conseguían con App Store en su primer fin de semana 10 millones de descargas [60] y en sus 8 primeros meses de funcionamiento 800 millones de descargas entre sus más de 25000 aplicaciones disponibles [61].

Al tiempo que se lanzaba App Store [62] HTC llegó con el primer teléfono en incorporar el sistema operativo Android de Google [63], con características muy similares a las ofrecidas por iPhone. Aunque los comienzos de cualquier plataforma son siempre inciertos y teniendo en cuenta que hasta estar bien desarrollada esta tecnología ninguno había podido con Apple, que seguían teniendo un éxito rotundo con todos sus lanzamientos, no paso mucho tiempo para que los fabricantes se hayan dado cuenta del auténtico potencial de esta plataforma. Actualmente es el sistema con el potencial de

desarrollo más importante en el mundo de la telefonía móvil [64] y está absorbiendo a grandes pasos la cuota de mercado [65].

El sistema operativo Android [66] basa su funcionamiento en Linux y en un principio se destinaba a dispositivos móviles, concretamente Smartphone, pero hoy en día su desarrollo está expandido para soportar otros dispositivos como tablets, reproductores MP3, netbooks, PC e incluso televisores.

Android fue desarrollado en el seno de una pequeña empresa, Android Inc., que más tarde fue comprada por el gigante Google en 2005 [67]. Y fue presentado como producto emblema [68] de la Open Handset Alliance [69], una asociación de fabricantes y desarrolladores de hardware, software y operadores de servicio, que en sus inicios contaba con 35 empresas y actualmente cuenta con 78.

Este sistema operativo tiene una gran comunidad de desarrolladores que, gracias a su empeño por extender la funcionalidad de los terminales con Android, han sobrepasado las 250.000 aplicaciones disponibles para la tienda de aplicaciones oficial de Android [70], sin contar con otras tiendas que también ofrecen aplicaciones. Esto puede deberse en gran medida al lenguaje en que se desarrollan estas aplicaciones, Java, un lenguaje bastante estandarizado y sencillo de programar.

Sin embargo sus inicios no fueron tan prometedores. En el segundo trimestre de 2009 la compañía Canalys estimaba que Android tenía el 2,8% de la cuota de mercado de teléfonos inteligentes a nivel mundial, en un periodo de tiempo muy similar en el que Appel había conseguido un 14% [71]. Sin embargo experimenta un importante aumento en la cuota de mercado, como refleja un informe de ComScore [72], realizado en los Estados Unidos, donde el estudio realizado durante tres meses, entre Noviembre de 2009 y Septiembre de 2010, refleja que a medida que salen nuevos terminales con este sistema operativo van acaparando cuota de mercado, pasando de los 3,8% a 9% de los Smartphone. En el siguiente estudio [73], que termina en Septiembre de 2010, Android tiene un 21.4% de los usuarios de Smartphone, lo que le convierte en un competidor muy fuerte.

Concretamente en mayo de 2010, Android superó en ventas a iPhone, según un informe publicado por el grupo NPD [74], en el que se indicaba que Android obtuvo un 28% de ventas de teléfonos inteligentes en el mercado de los Estados Unidos. Más tarde un nuevo informe [75] rebelaría que Android se situaba en la cabeza, con un 33% de las ventas, desbancando al sistema operativo de BlackBerry, que se encontraba en el primer puesto desde 2007.

Solo tres años como sistema operativo le bastaban a Android para llegar a una cifra de más 500.000 dispositivos activados por día a nivel mundial, como informaba SiliconNews en julio de 2011 [76].

Android ha revolucionado el mercado de los videojuegos portátiles y el del Smartphone a la vez. Ha conseguido integrar ambas funciones en un mismo terminal, y además mantener un buen sistema operativo.

Los juegos móviles han cambiado, evolucionado y madurado en un periodo de tiempo sorprendentemente corto, y hoy por hoy hacen las delicias de los consumidores con una amplia gama de videojuegos, modalidades y tecnologías que siguen creciendo cada año.

En 2011 se asiste a una nueva era de creatividad en el ámbito de los videojuegos gracias tanto a las producciones de las grandes compañías multinacionales como a los esfuerzos de innovación de los desarrolladores más pequeños. Después de medio siglo el concepto de videojuego se ha evolucionado con el tiempo para acabar convirtiéndose en un medio de entretenimiento transformado y adaptado a los medios y tecnologías actuales.

2.2. Valoración del mercado

Después de haber realizado una completo repaso a la historia de los videojuegos, haciendo una breve mención a su origen y su respectivo desencadenante de las videoconsolas como percutor del ocio que hoy en día conocemos, y en el que nos hemos centrado de una manera más extensa en la evolución del videojuego en los dispositivos móviles, llegamos a un punto en el que tenemos que hacer un balance de los videojuegos existentes que puedan servirnos de precedentes para el desarrollo del proyecto.

Si recordamos, los requisitos imprescindibles con los que este proyecto debía contar eran básicamente cuatro:

- Diseño de la lógica de un videojuego que permitiera el uso de las diferentes tecnologías presentes en los terminales Smartphone
- Uso del GPS incorporándolo en el videojuego como una de las bases imprescindibles para su desarrollo.
- Hacer uso de una librería creada para el uso de realidad aumentada en videojuegos Android en el proyecto de fin de carrera que se desarrolla paralelamente a éste.
- Uso de las librerías de Google como apoyo para la correcta consecución del juego.

Teniendo en cuenta las condiciones con las que el videojuego tiene que cumplir, se hará un análisis del mercado en pos de encontrar diferentes aplicaciones o videojuegos que puedan ayudar a entender las posibilidades de nuestro proyecto.

Al tratarse de un videojuego que se basa principalmente en el uso del GPS nos centraremos en aquellas aplicaciones o videojuegos que hagan un uso implícito de éste o de posicionamiento sobre mapas.

Pero antes de ello deberíamos hacer un breve análisis de los diferentes campos en los que el uso del sistema de posicionamiento global se puede aplicar. De esta manera se facilita la búsqueda de videojuegos que ayuden en el desarrollo de este proyecto fin de carrera.

2.2.1. Aplicaciones del GPS

Hoy en día el uso del sistema de posicionamiento global, GPS, está muy generalizado. Hacemos un uso cotidiano de él y aunque conocer nuestra posición pueda parecer algo trivial, se ha convertido en algo imprescindible en muchos aspectos de nuestras vidas, indistintamente de que su uso sea profesional o no. A grandes rasgos, podemos dividir los campos de aplicación en cinco [77].

Posicionamiento: quizás ésta sea la aplicación más evidente del GPS. El sistema de posicionamiento global es la primera herramienta capaz de determinar una posición o una localización a lo largo de todo el diámetro del planeta, con un mínimo error [78], bajo cualquier circunstancia.

Navegación: se trata de una de las aplicaciones más extendidas de uso cotidiano gracias a los navegadores de a bordo de los diferentes vehículos. Y es que, puesto que se pueden determinar dos posiciones pertenecientes a un lugar concreto, se pueden calcular diferentes trayectos o la mejor ruta entre esos dos lugares, ya sean para viandantes, conductores o capitanes de barcos [79].

Seguimiento: puede parecer una aplicación propia de una película, pero la verdad es que se trata de un uso bastante útil para localizar personas que, por ejemplo, tengan alguna enfermedad que les desoriente [80], con un dispositivo que comunica la posición a una central de seguimiento. Además, no se limita al seguimiento implícito de personas, sino que extiende su uso en campos como la aviación o el transporte ferrovial [81], para evitar colisiones, o en el seguimiento de fauna silvestre [82].

Topografía: gracias a la precisión del sistema, los topógrafos cuentan con una herramienta muy útil para la determinación de puntos de referencia, accidentes geográficos o infraestructuras [83], entre otros, lo que permite disponer de información topográfica precisa, sin errores y fácilmente actualizable.

Sincronización: gracias a los relojes atómicos que disponen los satélites [84] que establecen el posicionamiento se puede disponer de mediciones exactas en los receptores GPS. Esto nos permite emplear este sistema para determinar momentos en los que suceden o sucederán determinados eventos, sincronizarlos, unificar horarios, entre otras aplicaciones [85].

2.2.2. GPS en los móviles

Este apartado se centra en evaluar los diferentes campos de aplicación de GPS en busca de aplicaciones y videojuegos de la plataforma Android que puedan inspirar el videojuego que fundamenta las bases de este proyecto.

Entre los usos más comunes del GPS está, evidentemente, el de asistente de navegación que cualquier conductor puede tener en su coche. Todos tienen una dinámica parecida. Primero obtienen tu posicionamiento vía GPS, a partir de esta saben guiarte a cualquier punto que introduzcas, ya que tienen almacenados en una base de datos coordenadas correspondientes a las diferentes direcciones que puedas introducir, y a lo largo del trayecto, previamente programado y calculado para que cumpla una serie de pautas (trayecto más corto, más económico,…), te indica el camino a seguir. El resto cambia poco, mapas más coloridos, voces más cálidas, gráficos más vistosos, en definitiva cambios en el interfaz de usuario para crear identidad.

Los navegadores GPS más utilizados, bien por su calidad o por lo económico que resultan, por su apartado gráfico o por la cercanía con la que una máquina nos pueda tratar, entre otros, y dejando de un lado el universalmente conocido Google Maps, son *Copilot Live, AndNav2, iGO, Sygic* y *NDrive* [86].

Pero el sistema de posicionamiento global tiene otras muchas utilidades muy curiosas, aparte de las aplicaciones de navegación, que abren las posibilidades que esta herramienta nos brinda.

Por ejemplo GPS Essentials [87] es un programa que te permite usar diferentes funcionalidades del GPS como una brújula para orientarnos o monitorizar la cantidad de satélites que podemos captar, además de facilitarnos datos de distancias, velocidades o posiciones.

Car Locator [88] es una aplicación mucho más sencilla, pero bastante original, que nos evitará dar infinidad de vueltas por el barrio hasta encontrar donde dejamos aparcado nuestro coche, ya que te permite guardar la ubicación de tu coche una vez lo hayas aparcado.

Por otro lado encontramos una curiosa aplicación para amantes de la montaña, *Indet GPS* [89], que te permitirá planificar tus rutas, localizar por GPS las cimas que subes, grabar tu recorrido o cargar recorridos para que el GPS te muestre las rutas de otros usuarios.

Kulturize [90] es otra aplicación que gracias al posicionamiento GPS nos facilita

un listado, ordenados por la hora de inicio, de los eventos que podemos encontrar cerca de donde nos encontramos.

Hasta el momento hemos visto como hacer un uso bastante bueno del GPS, pero en ocasiones podemos encontrar aplicaciones que van más allá de todo esto. Basta con combinar el uso del posicionamiento global con algunas características de los móviles Smartphone, como pueden ser internet, la cámara u otros sensores de orientación.

Esto es lo que hacen aplicaciones como *Layar, TwittARound, Wikitude World Browser,* que haciendo uso de estas herramientas consiguen dar un enfoque de realidad aumentada a estas aplicaciones.

Layar

Layar [91] es un navegador de realidad aumentada que basa su funcionamiento en los datos que proporciona el GPS y la brújula de cualquier terminal inteligente, además de usar el sensor de movimiento [92] para saber hacia dónde está el usuario señalando su móvil. De esta manera la pantalla nos muestra lo que la cámara capta en esa dirección y sobre ella información relativa de lo que estamos viendo en ese mismo instante por el teléfono. Estas tecnologías se utilizan en conjunto para identificar la ubicación del usuario y el campo de visión.

A partir de la situación geográfica y la orientación, las diversas formas de datos se establecen en la vista de cámara con la inserción de capas adicionales, pudiendo visualizar puntos de interés, casas en venta, restaurantes, cajeros automáticos e incluso información introducida por otros usuarios [93]. Además esta aplicación nos permite conocer la distancia que nos separa de los puntos en cuestión, y se nos facilita una ruta a seguir a través de Google Maps hasta dicho punto de interés.

Puesto que Layar analiza y detecta la información disponible a tu alrededor, es necesario que se tenga activado tanto el GPS como la conexión a Internet del móvil.

TwittARound

TwittARound [94] se trata de una aplicación que permite observar todos los twits que se están publicando a tu alrededor en tiempo real de una manera muy rápida y precisa. Nuevamente se hace uso del GPS no solo para determinar tu localización sino, además la de los *Twitters* que tengas agregados.

Al tiempo que diriges la cámara del terminal a un lugar determinado, TwittARound superpone a las imágenes que visualizas en la pantalla unos pequeños iconos, que representan los *Twitters* que se encuentran en la dirección a la que se apunta el móvil, y en la zona inferior una pestaña que se puede desplegar para visualizar el historial de actualizaciones de dicho *Twitter* [95].

Wikitude World Browser

Wikitude World Browser es una aplicación móvil de realidad aumentada de Augmented Planet [96] que funciona como una enciclopedia y muestra infinidad de puntos de interés. Estos puntos de interés se logran con la interfaz de Google Maps y brinda a los usuarios datos sobre su entorno, gracias a la superposición de información que se conocen como Worlds [97], entre los que podemos encontrar ejemplos como: Flickr, Wikipedia, Youtube, Twitter, además de contenido generado por usuarios a los que brinda la posibilidad de configurar fuentes conocidas como MyWorld [98]. Todo esto se muestra en la pantalla del teléfono de manera muy similar a como lo hace Layar pero mostrando la información de una forma más ordenada y limitada.

Wikitude requiere brújula, acelerómetro, GPS y cámara [99]. Es gracias a la combinación de estas tecnologías que consigue hacer de la realidad aumentada una herramienta de información muy valiosa para todo tipo de personas.

Como se estado viendo existen aplicaciones muy útiles que hacen un uso muy eficiente de la realidad aumentada, integrando de una manera muy dinámica el uso del GPS, no solo ya como localizador de nuestra posición, sino también como indicador de comercios, restaurantes, situaciones de amigos…

Además hemos podido observar que muchas de estas aplicaciones hacían prácticamente, un uso intrínseco de Google Maps para la orientación de los usuarios en el uso de estas herramientas, lo que aumentaba su facilidad de uso.

Como se ha mostrado, estas aplicaciones dan un nuevo punto de vista al GPS que uniéndolo con otras herramientas, surgen novedosas utilidades y aplicaciones de la herramienta base de este proyecto.

Pero el GPS ha evolucionado hasta un nivel en el que se le da cabida en los videojuegos, para dotarlos de un dinamismo y una realidad fuera de lo normal. Esto comenzó a aplicarse en juegos como *ARQuake* o *Can You See Me Now?*

AR Quake

ARQuake [100] es una la versión de Realidad Aumentada del popular juego de Quake. Creado en el Laboratorio de Informática de la Universidad del Sur de Australia, ARQuake proporciona un shooter en primera persona que permite al usuario correr en el mundo real mientras juega a un juego en el mundo generado por ordenador. El sistema utiliza GPS, un sensor de orientación híbrido magnético e inercial y un ordenador portátil estándar llevado en una mochila. ARQuake fue el primer juego completo de trabajo de Realidad Aumentada creado para uso en exteriores.

Se modificó Quake para tomar la información de un sensor GPS que facilitara la localización del usuario y su orientación, y así cuando éste caminara, se estuviera

moviendo por el mundo de Quake, que se encontraba en sincronía con el mundo real. Los monstruos y los edificios parecen estar en el mundo real, como si te encontraras realmente allí. ARQuake básicamente permite al usuario jugar al famoso juego de Quake mientras está en el mundo físico real.

Can You See Me Now?

Una de las puestas en escena más representativas de aplicación del GPS a los juegos es el "Can You See Me Now?", de Blast Theory [101]. Es un juego on-line de persecución por las calles donde los jugadores empiezan en localizaciones aleatorias de una ciudad, llevando consigo un ordenador portátil con un receptor GPS. Como todo juego de persecuciones el principal objetivo es mantenerse alejado del perseguidor, por lo que los jugadores tendrán que evitar que un miembro de Blast Theory les alcance en un radio de menos de 5 metros, puesto que en este caso se les habrá capturado y pierden el juego.

Los ordenadores de mano permiten orientar a los perseguidores, el cual muestra la posición de los jugadores, tomando el tejido de la ciudad y haciendo que la ubicación del jugador sea el mismo centro dentro del juego. La primera edición tuvo lugar en Sheffield pero después se repitió en otras muchas ciudades europeas.

Can You See Me Now? da la posibilidad, puesto que se trata de un juego on-line, de jugar desde cualquier parte del mundo. De tal manera que los jugadores pueden jugar en línea en una ciudad virtual contra los miembros de Blast Theory, los cuales aparecen en sus ordenadores de mano con una posición ficticia en la ciudad virtual, aunque el desarrollo del juego siga llevándose en las calles de la ciudad de cada jugador. Los corredores seguidos por los satélites, pueden intercambiar tácticas y enviar mensajes a Blast Theory. Tiene una capacidad para 20 personas jugando en línea simultáneamente.

Pero estos dos videojuegos no dejan de ser dos aplicaciones de GPS pioneras, que hacían uso de herramientas muy sofisticadas [102], cuyas características son muy difíciles de encontrar en un Smartphone convencional.

Con todo se encuentran videojuegos como *GigaPutt*, *Spectrek* o *Zombies, Run!* los cuales hacen un uso del GPS, en la línea que de lo que se ha visto hasta ahora, con resultados bastante buenos.

GigaPutt

Se trata de un videojuego que convierte el barrio en el que el jugador se encuentra en un campo de golf [103]. Situando los diferentes hoyos a completar sobre una vista aérea del mismo, junto con diferentes extras como bocas de incendios que explotan o monedas que dan una puntuación extra. El lanzamiento de la pelota se hace haciendo uso del terminal, el cual se utiliza, gracias a los acelerómetros, como palo de golf.

GigaPutt es un juego que basa su funcionamiento en el sistema de posicionamiento global, para situar la ubicación del jugador, entre otros elementos, sobre una imagen satélite de Google Maps [104]. Además, tiene un modo multijugador, que permite jugar hasta tres amigos desde un mismo terminal.

Spectrek

Spectrek [105] es un videojuego disponible para Android y Iphone, que permite al jugador ponerse en la piel de un caza fantasmas. Para ello se hace uso del móvil, para poder identificar zonas en las que puede haber alguna actividad del más allá.

Gracias al GPS se pueden identificar las áreas en las que los fantasmas se encuentran, y andar hacia los espectros para capturarlos. Este juego te permite elegir entre diferentes modalidades cada una de ellas de duraciones distintas.

Zombies, Run!

Zombies, Run! [106] se trata de una buena propuesta para hacer ejercicio a la par que huyes, la mayor parte del tiempo, de una horda de zombis, los cuales te persiguen dentro del mapa que visualizas en el móvil.

Para poder jugar es necesario tener el GPS encendido, ya que gracias a este se controlará continuamente la posición del jugador sobre el mapa. Además gracias al GPS también se posicionarán los zombis sobre el mapa que se tendrá disponible.

Se trata de un videojuego bastante ingenioso que se ha visto modificado para llevarlo al límite en su nueva versión, la cual, una vez conseguida la financiación necesaria, saldrá a principios de 2012 para móviles inteligentes con sistema operativo de Appel y Android [107].

A esta nueva versión no solo se la ha dotado de una historia, sino que además se le ha dado un enfoque más deportivo, en el que tendrás que recorrer la ciudad en busca de objetos necesarios para la supervivencia, ayudado de unos auriculares los cuales serán los que te vayan indicando por donde moverte [108].

Capítulo 3

Análisis de las herramientas

En el punto anterior se han visto infinidad de aplicaciones y videojuegos que, la mayoría de ellos, tienen un uso cotidiano y que se puede ver en cualquier dispositivo normal en la calle, en parte gracias a la penetración de la telefonía móvil. Hoy podemos encontrar en manos de todo tipo de usuarios, incluso en aquellos que cabría pensar excluidos de las tecnologías más novedosas, terminales que proporciona un acceso constante a Internet, servicios basados en localización y ancho de banda masiva.

Pero podemos descubrir algunos casos aislados, como son los de *Can You See Me Now?* o *AR Quake*, que debido a las complejas prestaciones a las que son dirigidos, necesitan equipos mucho más sofisticados de lo que sería un móvil de última generación, con ratios de error mucho menores y localizaciones mucho más precisas. Concretamente en el desarrollo de *AR Quake* usaban un equipo que les proporcionaba errores del rango de 50 cm con 10 actualizaciones por segundo. Por lo que se hace necesario un estudio de las herramientas que se van a utilizar para el desarrollo de este proyecto.

Existen varios motivos que hacen necesario este estudio:

- Conocer las capacidades de las herramientas y su potencial, de esta manera se irá concibiendo el proyecto con una idea clara de las posibilidades que tiene a lo largo de su desarrollo.
- Centrar su uso en los dispositivos móviles. Ya que el potencial de alguna de las herramientas a nivel usuario difiere del que consiguen dispositivos especializados para un uso concreto, como es el caso del GPS. Se ha de conocer la precisión y los valores de orientación que un equipo de calidad estándar de un teléfono móvil suministra en un uso normal.
- Valorar las diferentes aplicaciones que tienen, ya habiendo hecho un estudio más concienzudo de las limitaciones de las herramientas. Lo que permitirá generar una idea preconcebida de lo que el proyecto podrá hacer.

Los dispositivos que se someterán a estudio serán los mismos que se instan a utilizar para acometer el proyecto. De esta manera nos encontramos con las siguientes herramientas:

- GPS. Se debe conocer el potencial de este instrumento y sus limitaciones

para integrarlo en la medida de lo posible en el proyecto como parte indispensable del mismo.

- API Google. El uso de estas librerías es otro de los requisitos impuestos por lo que se debe hacer un estudio exhaustivo de las funcionalidades que se podrán incluir al proyecto.

- Sistema Operativo Android. Se trata del sistema operativo funcional en gran parte de los Smartphone, se hace necesario dedicar algo de tiempo a desentrañar el funcionamiento de esta herramienta, las posibilidades que brinda y sus aplicaciones para el proyecto.

Así que este punto del documento recoge información relacionada con los diferentes instrumentos que se han de utilizar para el desarrollo del proyecto.

3.1. GPS (Sistema de posicionamiento global)

El GPS permite determinar en todo el mundo la posición de un objeto, una persona o un vehículo con una precisión hasta de centímetros con tecnología especializada. Se trata de un sistema que desarrolló, instaló y actualmente mantiene operativo el Departamento de Defensa de los Estados Unidos.

El GPS funciona mediante una red de 24 satélites en órbita a 20.200 km que cubren al completo la superficie de la Tierra. Cuando se desea conocer una posición, el receptor localiza como mínimo tres satélites, de los que recibe unas señales indicando la identificación y la hora del reloj de cada uno de ellos. Basándose en estas señales, el aparato sincroniza el reloj del GPS, con una exactitud extrema similar a la de los relojes atómicos a bordo de los satélites. Calcula el tiempo que tardan en llegar las señales al equipo, y determina la distancia de cada satélite respecto al punto de medición. Conocidas las distancias, se calcula fácilmente la propia posición relativa respecto a los tres satélites y se obtiene la posición absoluta o las coordenadas reales del punto de medición.

Teniendo información de un cuarto satélite, eliminamos el inconveniente de la falta de sincronización entre los relojes de los receptores GPS y los relojes de los satélites. Y es en este momento cuando el receptor GPS puede determinar una posición 3D exacta (latitud, longitud y altitud). Con todo, al no estar sincronizados los relojes entre el receptor y los satélites, la intersección de los cuatro satélites es un pequeño volumen en lugar de ser un punto exacto.

Además de los 24 satélites, que se encuentran repartidos en 6 planos orbitales de 4 satélites cada uno, y los receptores el sistema GPS cuenta además con estaciones terrestres, que envían información de control a los satélites para controlar las órbitas y realizar el mantenimiento de toda la constelación.

Debido al carácter militar del sistema GPS, el Departamento de Defensa de los EE. UU. se reservó la posibilidad de incluir un cierto grado de error aleatorio, que podía variar de los 15 a los 100 m. Pero esta premisa fue eliminada el 2 de mayo de 2000. Aunque actualmente no aplique tal error inducido, la precisión intrínseca del sistema GPS depende del número de satélites visibles en un momento y posición determinados.

Oficialmente posicionan cualquier objeto con una precisión aproximada de 15 metros, aunque en realidad un dispositivo GPS con un elevado número de satélites siendo captados, siempre que tengan una geometría adecuada, pueden obtenerse precisiones inferiores a 2,5 metros en el 95% del tiempo. Además si se cuenta con una tecnología DGPS, GPS diferencial, la precisión mejora hasta tal punto que es inferior a un metro en el 97% de los casos.

La posición calculada por un receptor GPS requiere el instante actual, la posición del satélite y el retraso medido de la señal recibida. La precisión es dependiente en la posición y el retraso de la señal.

Al introducir el retraso, el receptor compara una serie de bits procedente del satélite. Cuando se comparan los límites de la serie, las electrónicas pueden introducir una diferencia de un 1% del tiempo de BIT, aproximadamente 10 nanosegundos. Al propagarse las señales GPS a la velocidad de luz se presenta un error de 3 metros. Este es el error mínimo posible usando una señal convencional de GPS. Además de los ya mencionados hay que tener en cuenta otras posibles fuentes de error.

- Retraso de la señal en la ionosfera y la troposfera.
- Señal multirruta, producida por el rebote de la señal en edificios y montañas cercanos.
- Errores orbitales, donde los datos de la órbita del satélite no son completamente precisos.
- Errores locales en el reloj del GPS.

No obstante, el GPS está evolucionando hacia un sistema más sólido (GPS III), con una mayor disponibilidad.

Actualmente dentro del mercado de la telefonía móvil la tendencia es la de integrar, por parte de los fabricante, la tecnología GPS dentro de sus dispositivos. El uso y masificación del GPS está particularmente extendido en los teléfonos móviles Smartphone, lo que ha hecho surgir todo un ecosistema de software para este tipo dispositivos, así como nuevos modelos de negocios que van desde el uso del terminal móvil para la navegación punto a punto tradicional hasta la prestación de los llamados Servicios Basados en la Localización (LBS).

El sistema ha evolucionado y de él han derivado nuevos sistemas de posicionamiento IPS-2 (Inertial Positioning System), sistema de posicionamiento inercial, es un sistema de captura de datos que permite al usuario realizar mediciones a tiempo real y en movimiento, el llamado Mobile Mapping. Este sistema obtiene cartografía móvil 3D basándose en un aparato que recoge datos de un escáner láser, un sistema GNSS y un odómetro a bordo de un vehículo. Se consiguen grandes precisiones, gracias a las tres tecnologías de posicionamiento, dando la opción de medir incluso en zonas donde la señal de satélite no es buena.

3.2. API de Google

La documentación del API de Google está diseñada para personas que estén familiarizadas con la programación JavaScript y con conceptos de programación relacionados con objetos. Además nunca está de más, a la hora de estudiar el potencial de esta herramienta, estar familiarizado con Google Maps desde el punto de vista del usuario.

Esta documentación conceptualmente está diseñada para que cualquier persona pueda empezar a explorar y desarrollar rápidamente aplicaciones con el API de Google Maps. Se trata de un servicio gratuito, disponible para cualquier sitio web que sea gratuito para el consumidor.

La API de Google Maps se ha diseñado para que se cargue de forma rápida y para que funcione correctamente en los dispositivos móviles. En especial, se han centrado en el desarrollo para dispositivos móviles avanzados como, por ejemplo, iPhone y otros dispositivos móviles que ejecuten el sistema operativo Android. Los dispositivos móviles tienen tamaños de pantalla más pequeños que los navegadores habituales del escritorio. Asimismo, a menudo tienen un comportamiento determinado específico para estos dispositivos como, por ejemplo, "pinch-to-zoom", aplicar zoom al hacer clic en iPhone. Con su librería Google cuida todos estos detalles para el correcto funcionamiento en dispositivos móviles de última generación. Se hace especial hincapié en la habilitación de mapas rápidos y fiables en navegadores para dispositivos móviles.

Se ha implementado mediante una estructura MVC (Modelo Vista Controlador) modificada. Los cambios de estado de un objeto MVC como, por ejemplo, un mapa, se gestionan a través de definidores y captadores de un formato determinado. Asimismo, todos los estados de los objetos MVC se almacenan como propiedades de estos y la observación de los cambios de estado a través de gestores de eventos también tiene un formato determinado.

La documentación conceptual se divide en las áreas que aparecen a continuación

- Objetos de mapas básicos
- Eventos de mapas
- Controles de mapas
- Superposiciones de mapas
- Servicios de mapas

Para atender todas estas áreas, el API de Google dispone de una serie de clases

bastante completas y de conceptos sencillas de utilizar, que permiten al programador hacer un uso intuitivo y bastante eficiente para el desarrollo de la aplicación que desea implementar.

La librería de Google Maps está provista de clases bastante características y representativas de la funcionalidad que realizan. Además están estrechamente ligadas con conceptos utilizados en el desarrollo de aplicaciones móviles, como es la actividad, componente característico de una aplicación en Android. De esta manera nos podemos encontrar entre otras con GeoPoint, que nos permite identificar un punto geográfico mediante sus coordenadas, ItemizedOverlay, nos permite instanciar indicadores para colocar sobre nuestro mapa, MapActivity, MapController, MapView, que son clases referentes a los controles del mapa, la vista del mapa y, ya más concretamente, la actividad principal que manejará la aplicación, o MyLocationOverlay, la clase encarga de fijar el punto de referencia marcado por el localizador GPS.

Gracias a esta librería se pueden crear aplicaciones con funcionalidades tan variadas como las que a continuación se describen.

- **Google Directions API.** Permite a los usuarios solicitar indicaciones sobre cómo llegar mediante diferentes opciones de transporte.

- **Google Places API.** Ofrece información sobre empresas locales basada en su ubicación.

- **Maps JavaScript API.** JavaScript permite a los desarrolladores insertar un mapa de Google en sus páginas web, así como manipular el mapa y añadir contenido a través de diferentes servicios.

- **Maps API para Flash.** ActionScript API permite a los desarrolladores insertar un mapa de Google en sus aplicaciones o en sus páginas web basadas en Flash. Asimismo, les permite manipular el mapa en tres dimensiones y añadir contenido a través de diferentes servicios.

- **Google Earth API.** Permite a los desarrolladores insertar un globo digital en 3D en sus páginas web.

- **Static Maps API.** Permite a los desarrolladores insertar una imagen rápida y sencilla de Google Maps en sus páginas web o en sus sitios para móviles sin necesidad de utilizar JavaScript ni ningún sistema de carga de páginas dinámicas.

- **Servicios web.** Puedes utilizar las solicitudes de URL para acceder a información de lugares, de direcciones o de codificación geográfica de las aplicaciones cliente, y manipular los resultados en JSON o en XML.

- **Maps Data API.** Puedes visualizar, almacenar y actualizar datos de mapas a través de los feeds de Google Data API, mediante un modelo de funciones (marcadores, líneas y formas) y conjuntos de las mismas.

La ubicación geográfica hace referencia a la identificación de la posición geográfica de un usuario o dispositivo informático a través de diversos mecanismos de recopilación de datos. Normalmente, la mayoría de los servicios de ubicación geográfica utilizan direcciones de enrutamiento de red o dispositivos GPS internos para determinar esta ubicación. En la mayoría de los casos esta ubicación geográfica es un API específica de dispositivo en la que algunos navegadores o dispositivos son compatibles con ella, pero hay casos en los que no lo son, por lo que no se puede dar por hecho que la ubicación geográfica es posible para cualquier aplicación web.

Si se utiliza la API de Google Maps hay que indicar si la aplicación en la que se está trabajando utiliza un sensor para determinar la ubicación del usuario, por ejemplo, como ocurre en este proyecto, un localizador GPS. Esto resulta de especial importancia para los dispositivos móviles.

Las aplicaciones que determinan la ubicación de un usuario a través de un sensor deben transmitir *sensor=true* al cargar el código JavaScript del API de Google Maps. Y si el dispositivo no utilizara ninguna herramienta de detección hay que tener en cuenta que se deberá seguir transmitiendo este parámetro, estableciendo su valor en *false*.

3.3. Sistema Operativo Android

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework de desarrollo Java orientado a objetos. Aunque las aplicaciones son escritas en Java, no hay una Máquina Virtual de Java en la plataforma. El código Java no es ejecutable, se compila en el ejecutable Dalvik y corre en la Máquina Virtual Dalvik. Dalvik es una máquina virtual especializada diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados.

Se dispone de un entorno de desarrollo que hace el trabajo mucho más sencillo. El entorno de desarrollo incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software, todas estas herramientas se proporcionan con el SDK. El entorno de desarrollo integrado es Eclipse usando el plugin de Herramientas de Desarrollo de Android denominado ADT.

El desarrollo con Eclipse es el método aconsejado, ya que pueden invocar directamente las herramientas que usted necesita con Android Development Toolkit. Sin embargo, se puede optar por desarrollar con otro IDE o un editor de texto simple e invocar las herramientas por línea de comandos o con scripts.

Los pasos básicos para el desarrollo de aplicaciones con Eclipse son los siguientes:

- Crear Android Virtual Devices o conectar algún dispositivo de hardware. Es necesario crear dispositivos virtuales Android (AVD) o conectar dispositivos de hardware en el que se van a instalar las distintas aplicaciones que se desarrollen.

- Crear un proyecto de Android . Un proyecto de Android contiene todo el código fuente y archivos de recursos para su aplicación. Está construido en un paquete .apk que se puede instalar en cualquier dispositivo Android.

- Generar y ejecutar la aplicación . Si está utilizando Eclipse, el fichero ejecutable se generan cada vez que guarda los cambios y se puede instalar la aplicación en un dispositivo conectado o ponerlo en marcha en un AVD haciendo clic en **Ejecutar.**

- Depurar la aplicación con el SDK de la depuración y herramientas de registro . La depuración de la aplicación consiste en utilizar un depurador JDWP compatible con la depuración y registro de las herramientas que se proporcionan con el SDK de Android. Eclipse ya viene con un depurador

compatible.

- Probar la aplicación con la instrumentación disponible. El SDK de Android proporciona un marco de pruebas y la instrumentación necesaria para ayudar a establecer y ejecutar las pruebas dentro de un emulador o dispositivo.

Los componentes principales del sistema operativo de Android son:

- Aplicaciones: las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java y están compuestas por uno o más componentes de aplicaciones (actividades, servicios, proveedores de contenido y receptores de radiodifusión). Cada componente realiza una función diferente en el comportamiento global de la aplicación, y cada uno de ellos se puede activar de forma individual, aunque por otras aplicaciones. Existe un archivo que ha de contener toda aplicación, el manifiesto. En el archivo de manifiesto se tiene que declarar todos los componentes de la aplicación y también debe declarar todos los requisitos de aplicación, como la versión mínima de Android necesarios y todas las configuraciones de hardware necesarios.

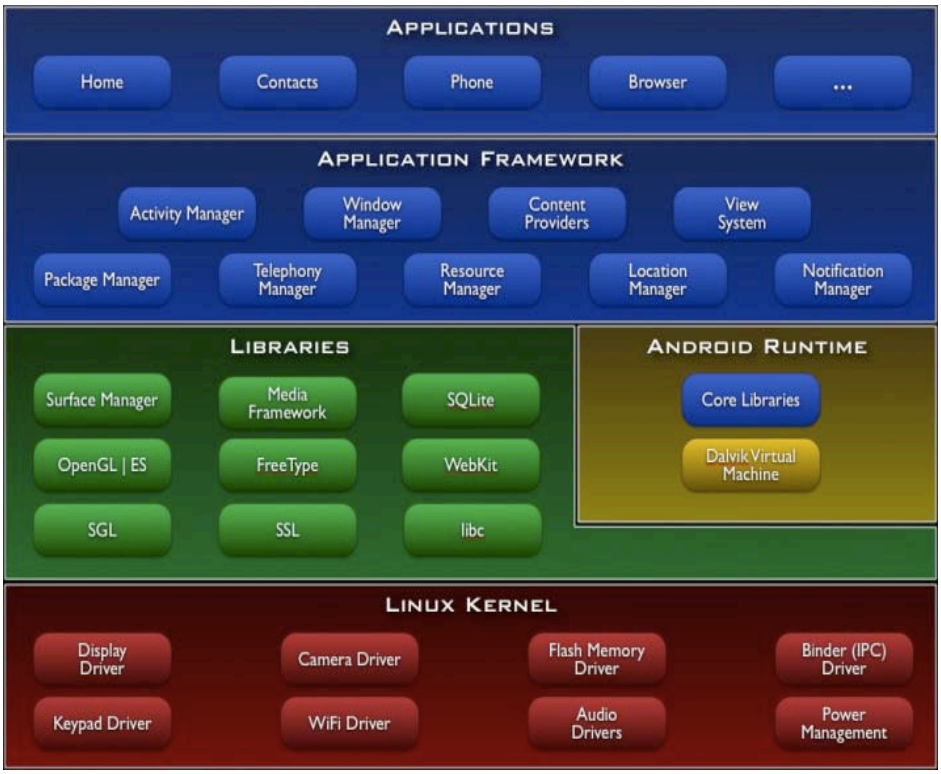
- Marco de trabajo de aplicaciones: al proporcionar una plataforma de desarrollo abierto, Android ofrece a los desarrolladores la capacidad de crear aplicaciones muy ricas e innovadoras. Los desarrolladores tienen la libertad recopilar información de los dispositivos de hardware, información de acceso a la ubicación, ejecutar servicios en segundo plano, establecer alarmas, añadir las notificaciones de la barra de estado, entre otras cosas. Los desarrolladores tienen pleno acceso a la API de un mismo marco utilizado por las aplicaciones básicas. La arquitectura de la aplicación está diseñada para simplificar la reutilización de componentes, y cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación podrá entonces hacer uso de esas capacidades (sujeto a restricciones de seguridad impuestas por el marco).

- Bibliotecas: Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android. Algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos 2D y 3D, SQLite, SGL y **FreeType**.

- Runtime de Android: Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato .dex por la herramienta incluida "dx".

- Linux Kernel: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

El siguiente diagrama muestra los componentes principales, ya explicados, del sistema operativo Android.



Android ha visto numerosas actualizaciones desde su liberación inicial. Estas actualizaciones al sistema operativo base arreglan bugs y agregan nuevas funciones. Generalmente cada actualización del sistema operativo Android está desarrollada bajo un nombre en código de un elemento relacionado con postres.

Es importante conocer las diferentes versiones en las que se puede desarrollar este proyecto, conocer las características que tiene cada una y las ventajas que tiene el utilizar uno u otra. Teniendo en cuenta que cuanto mayor sea la versión que se utilice, más restrictivo será el programa que se cree.

En la siguiente tabla podemos observar que mejoras han introducido las dos versiones más actuales, cuando se inicio el proyecto, entre las que surgió la versión elegida.

2.2 (Froyo) Basado en el kernel de Linux 2.6.32	• Optimización general del sistema Android, la memoria y el rendimiento • Mejoras en la velocidad de las aplicaciones, gracias a la implementación de JIT • Integración del motor JavaScript V8 del Google Chrome en la aplicación Browser • Soporte mejorado de Microsoft Exchange (reglas de seguridad, reconocimiento automático, GAL look-up, sincronización de calendario, limpieza remota) • Lanzador de aplicaciones mejorado con accesos directos a las aplicaciones de teléfono y Browser • Funcionalidad de Wi-Fi hotspot y tethering por USB • Permite desactivar el tráfico de datos a través de la red del operador • Actualización del Market con actualizaciones automáticas • Cambio rápido entre múltiples idiomas de teclado y sus diccionarios • Marcación por voz y compartir contactos por Bluetooth • Soporte para contraseñas numéricas y alfanuméricas • Soporte para campos de carga de archivos en la aplicación Browser • Soporte para la instalación de aplicación en la memoria expandible • Soporte para Adobe Flash 10.1 • Soporte para pantallas de alto número de Puntos por pulgada, tales como 4" 720p

2.3 (Gingerbread) **Basado en el kernel de Linux** **2.6.35.7 Actual en Smartphone**	• Actualización del diseño de la interfaz de usuario • Soporte para pantallas extra grandes y resoluciones WXGA y mayores • Soporte nativo para telefonía VoIP SIP • Soporte para reproducción de videos WebM/VP8 y decodificación de audio AAC • Nuevos efectos de audio como reverberación, ecualización, virtualización de los auriculares y refuerzo de graves • Soporte para Near Field Communication • Funcionalidades de cortar, copiar y pegar disponibles a lo largo del sistema • Teclado multi-táctil rediseñado • Soporte mejorado para desarrollo de código nativo • Mejoras en la entrada de datos, audio y gráficos para desarrolladores de juegos • Recolección de elementos concurrentes para un mayor rendimiento • Soporte nativo para más sensores (como giroscopios y barómetros) • Un administrador de descargas para descargar archivos grandes • Administración de la energía mejorada y control de aplicaciones mediante la administrador de tareas • Soporte nativo para múltiples cámaras • Cambio de sistema de archivos de YAFFS a ext4

Capítulo 4

Descripción del Videojuego

El capítulo anterior con sus diferentes puntos, evolución de la tecnología móvil, análisis del mercado y estudio de las herramientas disponibles, fue esencial para configurar la idea del videojuego que se va a implementar. Y es que sin este exhaustivo estudio no se hubiera podido imaginar el potencial de las herramientas de las que se dispone para el desarrollo del proyecto, ni tampoco la capacidad que nos brinda cada uno de estos instrumentos para realizar un juego realmente novedoso y competitivo en el mercado.

Sin haber afrontado el capítulo anterior se haría inevitable darse de bruces una y otra vez, y por mucho derroche de creatividad que hubiera inspirado este proyecto apenas se hubiera logrado algo que hubiera servido como punto de partida.

Llegado a un punto en que las ideas rondaban incesantes la cabeza, ante un sinfín de posibilidades que surgía de la convergencia de los juegos, Internet y los teléfonos móviles, se hacía necesario escoger una para iniciar el proyecto, coger una para darle forma y crear el videojuego objeto de este proyecto.

Este nuevo capítulo trata de definir bien el videojuego que se va a diseñar, surgido de la idea seleccionada, empezando por los objetivos, dentro del carácter lúdico que encierra todo juego, y pasando por la dinámica que el usuario ha de llevar a cabo para lograrlos.

No hay que olvidar que este videojuego se ha concebido con la finalidad de englobar otro proyecto fin de carrera desarrollado paralelamente a este y que es posible que todas las funcionalidades aquí descritas no se puedan llevar a cabo con la totalidad de este proyecto.

Además hay que tener en cuenta que el videojuego, aparte de cumplir los requisitos impuestos para este proyecto de fin de carrera, ha de integrar la funcionalidad que desarrolla el otro en el que se basa éste. Si recordamos los requisitos a cumplir en este trabajo estaban estrechamente relacionados con la implementación de la lógica de un videojuego, para dispositivos móviles Smartphone con sistema operativo Android, que permitiera integrar a la perfección las funcionalidades de un dispositivo GPS, disponibles en este tipo de terminales, y la utilización de las librerías de Google para mejorar la experiencia del usuario. Además, se deberá crear el interfaz que permitirá al usuario interactuar con el videojuego e implementar las funcionalidades propias de la librería que se desarrolla en el otro proyecto que representa el punto de partida de éste y que permite la adhesión de gráficos 3D a las imágenes captadas por la cámara y el uso

de los acelerómetros, lo que dará al videojuego un concepto aun mayor de realidad aumentada.

Está claro que cuando se pide que el videojuego haga uso de la localización GPS que un Smartphone nos facilita, se nos está pidiendo que nuestro videojuego interactúe de alguna manera con la realidad que nos rodea, se nos pide una fusión del mundo virtual en el que cualquier videojuego nos sumerge y el mundo al que estamos acostumbrados día a día. Algo que envolverá aún más a los usuarios del videojuego en la trama de este y que aumente la experiencia hasta puntos insospechados.

Teniendo en cuenta todo esto, una de las mejores opciones para generar un videojuego que englobe todas estas características es un videojuego basada en realidad aumentada. Esto permitirá al usuario involucrarse de una manera más activa en el videojuego, permitiéndole interactuar con la realidad, mejorando las expectativas que cualquier otro tipo de juego pudiera generar.

El juego que se ha ideado para atender estos requisitos consiste en situar al usuario en el centro de un mundo entre lo virtual y lo real, en el que sólo el uso de la cámara le servirá de nexo de unión con el mundo virtual.

El usuario caminará por el mundo real e interactuará con el mundo digital a través de la pantalla de su móvil que captará las mismas imágenes del mundo que le rodea, pero dejando entrever lo que la realidad esconde, vivimos en un mundo donde una serie de sujetos son androides.

Con estos propósitos se ideo el siguiente argumento:

El gobierno había encontrado la manera de realizar importantes estudios sociológicos implicándose de una manera bastante activa en estos. Había diseñado una inteligencia artificial lo suficientemente potente como para que, integrándolo en un prototipo humanoide, e induciéndolo algunos ideales, pudiera integrarse en los grupos sociales más radicales y peligrosos para observar y anotar conductas sociológicas, sin necesidad que esto conllevara un peligro.

En algún punto de uno de los estudios algo salió mal y ahora un grupo de androides con tendencias psicópatas deambula por las diferentes ciudades del mundo, que se encuentran vulnerables ante el desconocimiento de esta nueva amenaza.

El usuario se convierte en un agente secreto del gobierno al que se ha prometido un retiro de por vida sin ningún tipo de preocupaciones si logra, a toda costa, evitar que el fallo del gobierno salga a la luz. Para ello se le ha transportado al centro neurálgico en el que ocurrieron los hechos, será el lugar donde el usuario inicie el juego, y tendrá que identificar y capturar cuantos androides pueda antes de que alguno de ellos cometa su primer crimen, obligando al gobierno a publicar los sucesos e intentar enmendarlos con soluciones más drásticas.

Por suerte los movimientos de los androides están vigilados por satélites y nuestro agente cuenta con un dispositivo, el único en el mundo, que permite diferenciar a estos androides de personas corrientes, y además localizarlos en un mapa. Tendrá que hacer uso de su pericia y rapidez para intentar retrasar lo máximo un hecho imposible de ocultar.

Resulta un juego bastante dinámico en el que el estrés de una cuenta atrás precipita todos los sucesos, acelerando las acciones del usuario. Además con este videojuego se ha querido hacer un guiño al sistema operativo Android, que en sus inicios hacía lo mismo con la película *Blade Runner* con su teléfono *Nexus One*. Es más nuestro juego guarda muchas similitudes con esta película en la que Rick Deckard, un ex agente retirado de los Blade Runners, tiene que encontrar y acabar con unos replicantes fugitivos del modelo *Nexus 6*, seres fabricados a través de la ingeniería genética usados como esclavos en las colonias exteriores de la Tierra para realizar los trabajos más peligrosos.

Dejando de lado todas las peculiaridades del juego vamos a dar paso a una explicación más completa de lo que el videojuego hace y como atiende los diferentes requisitos impuestos.

Hasta ahora sabemos que el objetivo del juego es atrapar unos androides en un tiempo limitado. Estos androides serán un diseño en 3D, creado con la librería que el otro proyecto nos facilita, que más adelante explicaremos dónde, cuándo y por qué se podrán visualizar en el videojuego.

Básicamente se trata de conseguir atrapar, antes de que el tiempo finalice, cuantos más androides posibles mejor, ya que aumentarán la puntación final obtenida. El tiempo con el que cuenta el usuario son 15 minutos, prorrogables siempre y cuando se den una serie de circunstancias, como, por ejemplo, que coja un ítem que para el tiempo. La cuenta atrás estará siempre visible en la pantalla del dispositivo.

Si el contador de tiempo llega a cero sin haber capturado a todos los androides, el jugador habrá fracasado en su misión y será el final del juego. No se habrá logrado ocultar el error del gobierno y el ansiado retiro tras un último trabajo bien hecho tendrá que posponerse a que surja una nueva oportunidad, lo que se tarde en iniciar una nueva partida.

El juego constará de una serie de rondas, en las que habrá un número variable de androides, que crecerá con el paso de los sucesivos niveles. Evidentemente para pasar de ronda tendrás que capturar a todos los androides de ese nivel. Para capturar cada uno de los androides será necesario encontrarlo a través de la cámara del dispositivo y tocarlo en la pantalla del terminal, siempre que la distancia sea apropiada. Ya que solo se podrá identificar a un androide si te encuentras a una distancia menor de 20 metros, apareciendo en la pantalla del teléfono móvil, y sólo podrás capturarlo si te encuentras a

menos de 7 metros.

Capturar a un androide supone dos cosas. La primera que éste desaparezca y que por tanto quede un androide menos para pasar de ronda, a no ser que fuera el último, lo que supondrá el cambio automático de ronda. Y la segunda que se incremente el acumulado de puntos conseguidos, un extra del juego que refleja lo bien que se puede hacer y que permite compararlo con puntaciones obtenidas en partidas anteriores.

Si se logra pasar con éxito cada una de las rondas que van surgiendo, que ascienden a un total de diez, el jugador habrá logrado terminar el juego, lo que además de suponer una bonificación extra de puntos para la puntuación final que quedará reflejada en un ranking, conseguirá ese ansiado retiro que el gobierno le había prometido.

Evidentemente los androides no esperarán en un sitio fijo a que les captures. Estos se estarán moviendo de un lado a otro como transeúntes normales, lo que dificultará un poco más la misión del usuario de este videojuego.

En principio se pensó en dejar el número de rondas abierto y que el usuario pudiera disfrutar de niveles elevados donde el dinamismo del juego rozará lo excéntrico, pero finalmente se decidió darle un límite de rondas para que el juego pudiera tener un final feliz.

Con el paso de cada nivel se incrementará el número de androides que capturar, por lo que la dificultad del juego va aumentando. Pero con la sucesión de rondas existe una bonificación de tiempo extra por nivel completado, de tal manera que el tiempo que se tiene para capturar a los androides de la siguiente ronda sea el que te sobró de la anterior más el que se sumó por bonificación de ronda completada. Este tiempo será siempre el mismo, cinco minutos. Además de una bonificación de tiempo extra también se obtendrá una bonificación de puntos.

Si el juego no proporcionara más ayuda que la bonificación extra de tiempo por ronda completada, este juego se haría bastante complicado, pero, aparte de esta bonificación, el juego está provisto de una serie de extras que facilitan la misión del jugador.

Al estar los movimientos de los androides vigilados por satélites, el juego cuenta con un mapa de la zona en la que se encuentra el jugador, y en el que se señala la posición del usuario y de los androides. De esta manera los androides están siempre localizados y no resulta tan difícil encontrarlos. Al iniciar una nueva partida, y al pasar de ronda, el mapa toma como centro la posición del usuario del juego.

Además de este mapa, ocasionalmente y de manera aleatoria aparecerán marcados en el mapa, una serie de ítems (objetos) que ayudarán temporalmente al jugador. La manera de obtener los beneficios que cada uno aporta es similar a la

captura de un androide. Tienes que localizarlo con el terminal y pulsarlo en la pantalla del mismo estando a una distancia apropiada.

A continuación se describe el funcionamiento de estos:

- Parar el tiempo. Como ya hemos comentado con anterioridad, existe un ítem que permite detener la cuenta atrás, de tal forma que el jugador tenga unos segundos más para cumplir su misión. La duración de este efecto será de unos 45 segundos.

- Congelar androides. Este ítem permite paralizar temporalmente a todos los androides de una misma ronda, de esta manera el jugador goza de una ventaja adicional para completar la ronda durante un tiempo de unos 45 segundos. Este ítem puede parecer muy similar al anterior pero en realidad da una ventaja mayor puesto que al parar el tiempo los androides siguen en movimiento y estos son bastante resbaladizos.

- Puntos x2. Si durante el tiempo que dura el efecto de este ítem el usuario realiza alguna acción que conlleve la suma de puntos al contador, este incrementará el valor de la acción sumándose el doble del valor original de ésta. El efecto temporal de este ítem asciende al minuto y medio.

Todos los objetos, una vez que aparecen en el mapa, tienen un tiempo límite de 30 segundos para encontrarlos y recogerlos. El tiempo que tienes para disfrutar de sus beneficios como hemos podido comprobar varía en función del ítem.

La intención era tener una diversidad de objetos mucho mayor que se alejara de la monotonía y dotara al videojuego de un carácter más lúdico, pero debido a que se trataba de contenido extra que se añadía con objeto de acercar la aplicación a lo que se entiende por videojuego, se decidió implementar las mínimas funcionalidades.

Una vez definido el videojuego que servirá para desarrollar el proyecto integrando el otro que se ha desarrollado paralelamente, cabe hacer una mención de las partes que afrontará con más énfasis el proyecto que esta memoria expone.

Concretamente las funcionalidades que abarca y desarrolla este proyecto son:

- El desarrollo de toda la lógica del juego. Esto abarca diferentes aspectos como la inteligencia artificial de los androides, el número de androides que hay por ronda y el paso de ronda, las bonificaciones de las que se beneficia el usuario, la aparición de los ítems y el control de sus efectos, además de otros contenidos extras.

- El manejo del GPS para obtener la localización del usuario al inicio del videojuego, con lo que todo esto conlleva. Además del manejo de la

precisión del GPS y el control del error que este introduce. Evitando, en un principio, el inicio del juego hasta que no exista un precisión lo suficientemente aceptable y, durante el desarrollo de la partida, la perdida de precisión que en ocasiones se torna intolerable.

- El manejo del API de Google Maps para generar una vista de un mapa de la zona en el que el jugador se encuentra. Además, para enriquecer la partida con la librería de Google, se permite el manejo de diferentes funcionalidades como la de posicionar los androides, moverlos por el mapa y quitarlos, de igual manera que se puede hacer con los ítems.

- Por último, integrar la librería que se implementa en el proyecto de fin de carrera que se desarrolla paralelamente a éste, y que se usará para la introducción en este proyecto de los sensores de orientación, los acelerómetros y la inclusión de gráficos 3D, además de efectos de audio y control de eventos táctiles sobre la pantalla, todos ellos desarrollados en dicho proyecto.

Capítulo 5

Desarrollo

Hasta el momento hemos visto la parte del proyecto que abarca todos los aspectos que se podrían considerar como introducción, aportando los conocimientos y herramientas necesarias para llevar a cabo un desarrollo completo; que en este caso es la elaboración de un juego.

Planteada la idea en la que se basará el desarrollo del videojuego, se hará una descripción exhaustiva de la implementación de este proyecto, pero antes hay que pasar por las fases previas a este desarrollo.

En primer lugar se va a realizar la fase de análisis especificándose los requisitos. Otra parte importante del análisis, antes de modelar el concepto en los distintos objetos que se puede descomponer, consiste en analizar los estados en los que se pueden encontrar estos y cómo transitar de unos a otros y así conseguir pasar a la elaboración de diagramas de casos de uso.

Una vez esté analizada y definida la idea se pasará al siguiente punto, la fase de diseño, que abarcará cada elemento por separado de la estructura y lo acercará más al desarrollador; aunque sin llegar todavía a la implementación.

5.1. Análisis

Este punto se centra en la fase de análisis, para comenzar la implementación del proyecto. Para ello se parte de una necesidad de la que se desprenderán todos las necesidades que implican la realización de este proyecto, se irán tomando decisiones aplicables a cualquier proyecto real.

El propósito de este punto consiste en la identificación de los requisitos de usuario y los requisitos software. Una descripción del sistema a desarrollar a nivel de capacidades, restricciones, características del usuario, entorno operacional y dependencias.

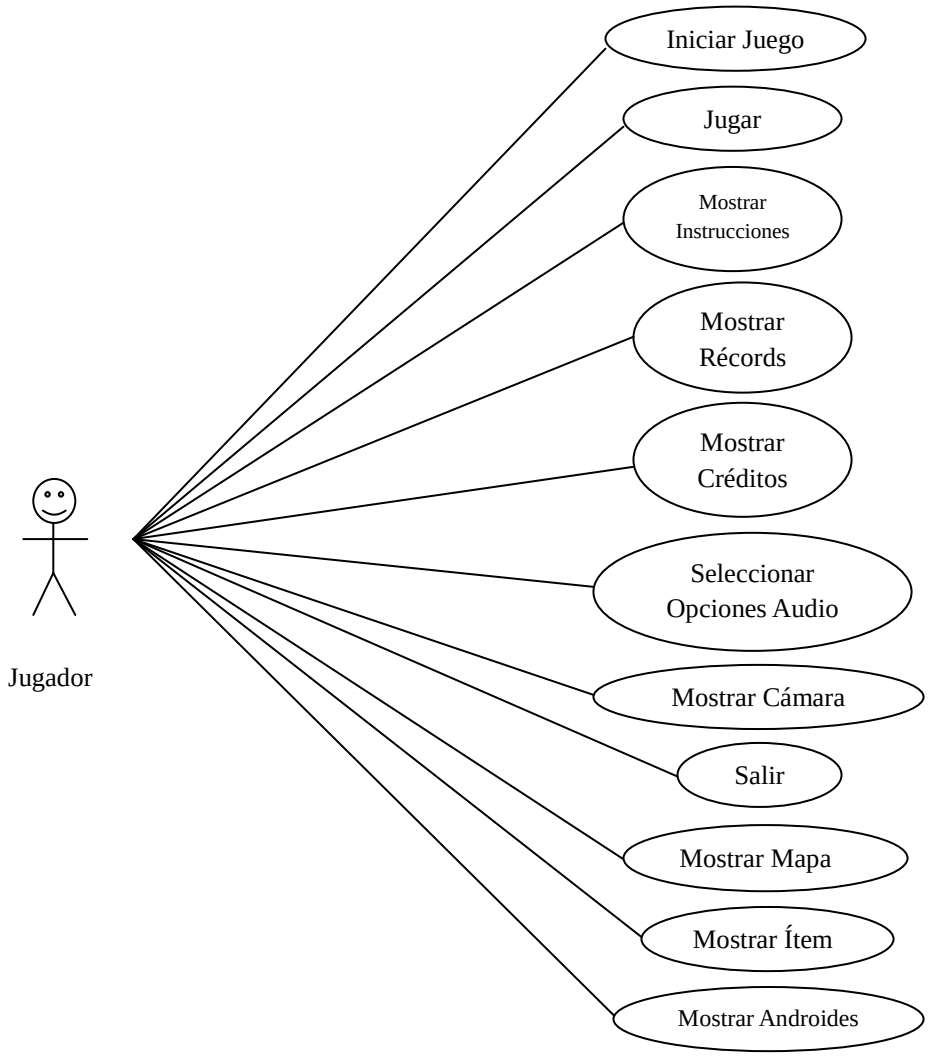
5.1.1. Casos de uso

Antes de exponer los requisitos de la aplicación, se deben elaborar los distintos diagramas de casos de uso resultantes de los requisitos definidos, así como la especificación textual de cada caso de uso para su mejor entendimiento, lo que nos permitirá determinar el origen de los requerimientos de usuario y software.

Un caso de uso es una secuencia de acciones que se dan entre un sistema y sus actores como respuesta a un evento que inicia el actor sobre dicho sistema.

En primer lugar se presenta el diagrama de casos de uso, el cual muestra de una manera sencilla los límites del sistema y cómo se utiliza. Sirve como una herramienta de comunicación que resume el comportamiento de un sistema y de sus actores. En nuestro caso sólo tenemos un actor, el jugador. Posteriormente se analizan los diferentes casos de uso, especificando en cada caso los actores, las precondiciones, las pos-condiciones, el escenario principal, y los escenarios alternativos y de excepción cuando sean precisos.

En los casos de uso se especifica qué hace el sistema en respuesta a una interacción de un usuario externo, por tanto no se tendrá en cuenta al propio sistema como actor (sólo sistemas externos que interactúen con el propio se pueden considerar actores). De esta forma, y por la naturaleza propia de algunas funcionalidades típicas de los videojuegos, surgirán casos de uso que quizá parezcan propios del sistema.



CU01: Iniciar Juego

Actores

- Jugador.

Precondiciones

- Haber instalado el juego.

Postcondiciones

- Se muestra la pantalla del menú principal del juego.

Escenario Principal:

1. El usuario selecciona el icono del juego en su dispositivo móvil.
2. El sistema muestra la pantalla del menú principal de opciones.
3. El caso de uso finaliza.

CU02: Jugar

Actores

- Jugador.

Precondiciones

- Haber iniciado el juego y mostrar el menú principal de opciones.

Postcondiciones

- - El jugador comienza la partida.

Escenario Principal:

1. El usuario selecciona la opción de Jugar.
2. El sistema inicia el temporizador que indica el tiempo de juego disponible.
3. El sistema muestra la pestaña de Mapa. (Ir al Caso de Uso 9: Mostrar Mapa)
4. El sistema muestra los robots que el usuario tiene que cazar en el nivel actual. (Ir al Caso de Uso 10: Mostrar Androides)
5. El usuario selecciona la pestaña de Cámara.
   (Ir al Caso de Uso 7: Mostrar Cámara)

Escenarios Alternativos:

*a. El tiempo de juego ha terminado
   1. El sistema disminuye el tiempo.
   2. El sistema verifica que se ha terminado el tiempo.
   3. El sistema registra la puntuación del usuario y se la muestra.
   4. El caso de uso finaliza.

*b. El usuario modifica su localización.

*c. El usuario puede seleccionar la pestana de Mapa
   (Ir al Caso de Uso Mostrar Mapa)

*d. El usuario puede seleccionar la opción "Atrás"
   1. El usuario selecciona el botón Atrás.
   2. El sistema finaliza el juego actual y muestra el menú principal.
   3. (Ir al Caso de Uso 1: Iniciar Juego)
   4. El caso de uso finaliza

*f. El usuario ha cazado todos los robots del nivel
   1. El sistema le indica al usuario que ha pasado al siguiente nivel.
   2. Ir al paso 3.

Escenarios de Excepción:

*e. Se pierde la señal GPS
   1. El móvil pierde la señal GPS.
   2. El sistema muestra un mensaje de aviso al usuario.

CU03: Mostrar Instrucciones

Actores

- Jugador.

Precondiciones

- Haber iniciado el juego y mostrar el menú principal de opciones.

Postcondiciones

- Se muestran las instrucciones correspondientes al videojuego.

Escenario Principal:

1. El usuario selecciona el botón Instrucciones del menú principal de opciones.
2. El caso de uso finaliza.

CU04: Mostrar Récords

Actores

- Jugador.

Precondiciones

- Haber iniciado el juego y mostrar el menú principal de opciones.

Postcondiciones

- Se muestran las puntuaciones más altas conseguidas en el videojuego.

Escenario Principal:

1. El usuario selecciona el botón Récords del menú principal de opciones.
2. El caso de uso finaliza.

CU05: Mostrar Créditos

Actores

- Jugador.

Precondiciones

- Haber iniciado el juego y mostrar el menú principal de opciones.

Postcondiciones

- Se muestran los créditos correspondientes al videojuego.

Escenario Principal:

1. El usuario selecciona el botón Créditos del menú principal de opciones.
2. El caso de uso finaliza.

CU06: Seleccionar Opciones Audio

Actores

- Jugador.

Precondiciones

- Haber iniciado el juego y mostrar el menú principal de opciones.

Postcondiciones

- Se activan o desactivan los efectos sonoros.

Escenario Principal:

1. El usuario selecciona el botón Opciones audio del menú principal de

opciones.

2. El usuario selecciona una de las dos opciones disponibles.
3. El caso de uso finaliza.

Escenarios Alternativos

*a. El usuario puede seleccionar la opción "Atrás"

1. El usuario selecciona el botón Atrás.
2. El sistema muestra el menú principal.
(Ir al Caso de Uso 1: Iniciar Juego)
3. El caso de uso finaliza.

CU07: Mostrar Cámara

Actores

- Jugador.

Precondiciones

- Haber seleccionado la opción Jugar del menú de opciones.

Postcondiciones

- Se muestra la imagen de la cámara.

Escenario Principal:

1. El usuario selecciona la pestaña de cámara.
2. El sistema muestra la imagen capturada por la cámara.

Escenarios de Excepción:

*e. Se pierde la señal GPS

1. El móvil pierde la señal GPS.
2. El sistema muestra un mensaje de aviso al usuario.

CU08: Salir

Actores

- Jugador.

Precondiciones

- Haber iniciado el juego y mostrar el menú principal de opciones.

Postcondiciones

- Se sale de la aplicación del juego.

Escenario Principal:

1. El usuario selecciona el botón Salir del menú principal de opciones.
2. El caso de uso finaliza.

CU09: Mostrar Mapa

Actores
- Jugador.

Precondiciones
- Iniciar la partida.

o
- Haber iniciado la partida y, encontrándose en la pantalla de la *Cámara,* seleccionar la pestaña de *Mapa.*

Postcondiciones
- El jugador puede visualizar un mapa de la zona en la que se encuentra.

Escenario Principal:
1. El sistema muestra por defecto el mapa al iniciar la partida.

Escenarios Alternativos:

*a. El usuario se encuentra en la pestaña de Cámara
1. El usuario pulsa la pestaña de mapa.
2. El sistema muestra el mapa.
3. El caso de uso finaliza.

*b. El usuario puede seleccionar la opción “Atrás”
1. El usuario selecciona el botón Atrás.
2. El sistema finaliza el juego actual y muestra el menú principal.
3. El caso de uso finaliza.

Escenarios de Excepción:

*e. Se pierde la señal GPS
1. El móvil pierde la señal GPS.
2. El sistema muestra un mensaje de aviso al usuario.

CU10: Mostrar Androides

Actores
- Jugador.

Precondiciones
- El usuario ha de haber iniciado la partida.
- Se tiene que haber cargado el mapa.
- Ha de haber una precisión del GPS aceptable.

o
- Haber completado una ronda con exito

Postcondiciones
- El jugador puede visualizar en el mapa los androides que puede capturar.

Escenario Principal:
1. El sistema carga el mapa y a continuación genera aleatoriamente las posiciones de los robots.

Escenarios Alternativos:

*a. El usuario completa una ronda
1. El sistema muestra el mensaje de ronda completada.
2. El sistema carga el mapa nuevo.
3. Se generan las nuevas posiciones correspondientes a los robots de la nueva ronda
4. El caso de uso finaliza.

*b. El usuario puede seleccionar la opción “Atrás”
1. El usuario selecciona el botón Atrás.
2. El sistema finaliza el juego actual y muestra el menú principal.
3. El caso de uso finaliza.

Escenarios de Excepción:

*e. Se pierde la señal GPS
1. El móvil pierde la señal GPS.
2. El sistema muestra un mensaje de aviso al usuario.

CU11: Mostrar Ítems

Actores

- Jugador.

Precondiciones

- El usuario ha de haber iniciado la partida.
- Se tiene que haber cargado el mapa.
- Ha de haber una precisión del GPS aceptable.
- Ha de haber pasado un tiempo establecido para generar ítems.

Postcondiciones

- El jugador puede visualizar en el mapa el ítem generado por el sistema.

Escenario Principal:

1. El usuario consume un tiempo determinado y el sistema genera aleatoriamente un ítem que coloca sobre le mapa.

Escenarios Alternativos:

*a. El usuario puede seleccionar la opción “Atrás”
1. El usuario selecciona el botón Atrás.
2. El sistema finaliza el juego actual y muestra el menú principal.
3. El caso de uso finaliza.

Escenarios de Excepción:

*e. Se pierde la señal GPS
1. El móvil pierde la señal GPS.
2. El sistema muestra un mensaje de aviso al usuario.

5.1.2. Requisitos de Usuario

Los requisitos de usuario reflejan las necesidades que debe cubrir el sistema desde el punto de vista del usuario. A partir de estos requisitos se podrá entender lo que espera el usuario del sistema que construir y llevarlo a cabo con éxito. Dentro de los de usuario se distinguen dos tipos: **requisitos de capacidad**, que representan una necesidad o servicio requerida por el usuario. Y **requisitos de restricción**, que son las restricciones impuestas por los usuarios sobre cómo se debe resolver el problema o cómo se debe alcanzar un objetivo.

RU 01 - Iniciar Juego			
Descripción	Seleccionar el icono del juego desde el menú de opciones del móvil.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RU 02 - Mostrar Menú			
Descripción	Mostrar menú de opciones antes de comenzar una partida.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RU 03 - Jugar			
Descripción	Comenzar una nueva partida.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RU 04 - Mostrar pestañas disponibles			
Descripción	Durante el juego, el usuario podrá elegir, mediante el uso de pestañas, visualizar la imagen del mapa o la imagen de la cámara.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RU 05 - Visualizar robot a capturar			
Descripción	El usuario podrá visualizar sobre la imagen de la cámara los robots que debe capturar.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RU 06 - Visualizar ayudas			
Descripción	El usuario podrá visualizar sobre la imagen de la cámara las ayudas que puede coger.		
Origen	Usuario	**Verificable**	Si
Necesidad	Opcional	**Claridad**	Si
Prioridad	Media	**Estabilidad**	Media

RU 07 - Capturar robot/ayudas			
Descripción	El usuario podrá capturar los robots y las ayudas mediante la pantalla táctil del móvil.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RU 08 - Activar / Desactivar audio			
Descripción	Poder activar o desactivar los efectos de sonido antes de comenzar una nueva partida.		
Origen	Usuario	**Verificable**	Si
Necesidad	Opcional	**Claridad**	Si
Prioridad	Baja	**Estabilidad**	Alta

RU 09 - Mostrar Instrucciones			
Descripción	Mostrar las instrucciones del juego.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Media	**Estabilidad**	Alta

RU 10 - Mostrar créditos			
Descripción	Mostrar los créditos del juego.		
Origen	Usuario	**Verificable**	Si
Necesidad	Opcional	**Claridad**	Si
Prioridad	Baja	**Estabilidad**	Alta

RU 11 - Mostrar récords			
Descripción	Mostrar las 3 puntuaciones más altas obtenidas en el juego.		
Origen	Usuario	**Verificable**	Si
Necesidad	Opcional	**Claridad**	Si
Prioridad	Baja	**Estabilidad**	Media

RU 12 - Salir			
Descripción	Cerrar el juego.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RU 13 – Seleccionar pestaña mapa			
Descripción	El usuario podrá seleccionar la pestaña del mapa que desplegará un mapa de la zona en la que se encuentra.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RU 14 – Visualizar robot en mapa			
Descripción	El usuario podrá visualizar sobre la imagen del mapa los robots que debe capturar.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RU 15 – Visualizar ayudas en mapa			
Descripción	El usuario podrá visualizar sobre la imagen del mapa las ayudas que podrá recoger.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RU 16 – Completar ronda			
Descripción	El usuario podrá completar una pasar de ronda una vez capture todos los robots de la ronda en la que se encuentra.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RU 17 – Finalizar juego			
Descripción	El usuario podrá terminar el juego si logra completar con éxito las 10 rondas disponibles.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RU 18 – Terminar partida			
Descripción	El usuario terminará la partida si agota el tiempo disponible para afrontar los diferentes niveles.		
Origen	Usuario	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

5.1.3. Requisitos Software

Dentro de los requisitos software, se encuentra los **requisitos funcionales** que describen lo que debe hacer el sistema, y los **requisitos de rendimiento**, **interfaz**, **operación**, **recursos**, **comprobación**, **documentación**, **seguridad**, **calidad**, **mantenimiento**, **daño** y **aceptación de pruebas** que limitan la forma en que debe llevarse a cabo.

Estos requisitos tienen un alto grado de importancia en cuanto a verificación y seguimiento del juego, ya que este deberá cubrir todos y cada uno de los requisitos identificados para afirmar que está completo.

RSF 01 – Generar mapa			
Descripción	El sistema generará una vista de un mapa de Google.		
Origen	RU 13	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RSF 02 – Obtener localización usuario			
Descripción	El GPS obtiene la localización del usuario para poder centrar el mapa en la zona en la que le usuario se encuentra.		
Origen	RU 13	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RSF 03 – Generar nivel			
Descripción	El sistema genera el posicionamiento aleatorio de los robots que se tendrán que capturar para completar el nivel.		
Origen	RU 03 o RU 16	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RSF 04 - Mostrar robot sobre el mapa			
Descripción	El sistema coloca sobre el mapa los marcadores de posición de los robots, a partir de las coordenadas generadas con anterioridad.		
Origen	RU 14	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RSF 05 – Generar ayuda			
Descripción	El sistema genera un ítem aleatoriamente, dotándolo de una posición y una funcionalidad.		
Origen	RU 15	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RSF 06 – Mostrar ayuda sobre el mapa			
Descripción	El sistema coloca sobre el mapa el marcador de posición del ítem, a partir de las coordenadas generadas con anterioridad.		
Origen	RU 15	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Media	**Estabilidad**	Media

RSF 07 – Borrar robot			
Descripción	El marcador que señala un robot determinado se borra del mapa, porque ha sido capturado.		
Origen	RU 07	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RSF 08 – Borrar ayuda			
Descripción	El marcador que señala un ítem determinado se borra del mapa, bien porque se ha recogido o bien porque el tiempo para intentar recogerlo ha expirado.		
Origen	RU 07	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Baja	**Estabilidad**	Alta

RSF 09 – Mover Robots			
Descripción	El sistema actualiza la posición de los robots para simular su desplazamiento		
Origen	RU 03	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Media	**Estabilidad**	Alta

RSF 10 – Detener tiempo			
Descripción	El sistema detiene el tiempo tras conseguir el ítem de parar el tiempo.		
Origen	RU 07	**Verificable**	Si
Necesidad	Opcional	**Claridad**	Si
Prioridad	Baja	**Estabilidad**	Alta

RSF 11 – Congelar robots			
Descripción	El sistema detiene el movimiento de los robots tras conseguir el ítem de congelar robots.		
Origen	RU 07	**Verificable**	Si
Necesidad	Opcional	**Claridad**	Si
Prioridad	Baja	**Estabilidad**	Media

RSF 12 – Multiplicar puntuación x2			
Descripción	El sistema multiplicará por dos todas las puntuaciones conseguidas hasta que expire el efecto de la ayuda tras conseguir el ítem de puntuación x2.		
Origen	RU 07	**Verificable**	Si
Necesidad	Opcional	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RSF 13 – Actualizar puntuación			
Descripción	El sistema actualizará la puntuación conseguida a lo largo del juego por la captura de androides.		
Origen	RU 07 o RU 16	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RSF 14 – Avisar de ronda completada			
Descripción	El sistema mostrará una pantalla indicando que un nivel se ha completado con éxito.		
Origen	RU 16	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RSF 15 – Avisar de juego completado			
Descripción	El sistema mostrará una pantalla indicando que el juego se ha completado con éxito tras haber superado las diez rondas disponibles.		
Origen	RU 17	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

RSF 16 – Avisar de fin del juego			
Descripción	El sistema mostrará una pantalla indicando que el juego se ha terminado si el contador disponible para completar los sucesivos niveles se agota.		
Origen	RU 18	**Verificable**	Si
Necesidad	Esencial	**Claridad**	Si
Prioridad	Alta	**Estabilidad**	Alta

5.2. Diseño conceptual

Para poder llevar el desarrollo del videojuego, hay que saber qué módulos lo componen, cómo se relacionan éstos, y en nuestro caso, cuales pertenecen a las competencias de este proyecto y cuáles no.

En la fase de diseño se debe tomar el análisis de la etapa anterior y realizar un diseño exhaustivo de los componentes, para que después, en la etapa posterior de implementación, esté todo tan claro como para no tener que replantear ningún aspecto. Por tanto en esta fase, hay que centrarse en obtener un diseño que se adapte a las necesidades de la aplicación y una definición exhaustiva de las clases, así como su respectivo diagrama que sirva para ver la colaboración e interacción de éstas.

Una vez estudiado los requisitos funcionales del sistema, las funcionalidades y el comportamiento del sistema, mediante el análisis en el punto anterior, se está en disposición de comenzar a definir los diferentes módulos que compondrán el modelo de diseño, de forma que cubran las necesidades y funcionalidades del software.

5.2.1. Arquitectura del sistema

En primer lugar, se deben identificar los módulos en los que se descompone el proyecto, y cómo éstos se relacionan entre sí. En nuestro caso, podemos diferenciar módulos de primer nivel y módulos de segundo nivel, adquiriendo los primeros mayor relevancia que los segundos. En el primer grupo encontramos los siguientes módulos:

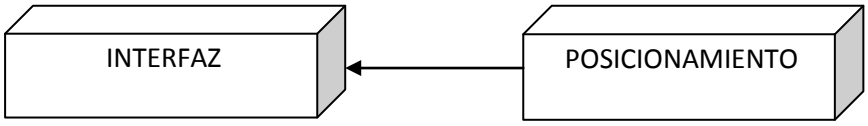
- Interfaz gráfica
- Posicionamiento
- Sensores
- Lógica
- Gráficos

y en el segundo:

- Eventos de pantalla
- Sonido
- Persistencia

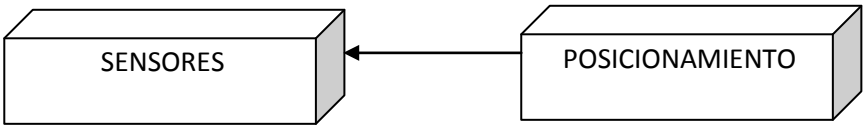
Una vez identificados los diferentes módulos, debe estudiarse cómo se interaccionan entre sí.

Interacción 1



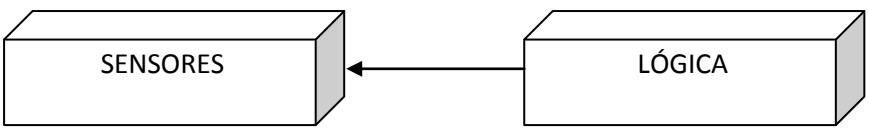
Como bien se ha comentado en los requisitos, cuando se inicie una nueva partida del videojuego, se mostrará un mapa de la zona en la que se encuentra el usuario, mostrando la posición de éste y de los robots a capturar. Por tanto, el módulo interfaz mostrará el mapa proporcionado por el módulo de posicionamiento.

Interacción 2



El módulo de sensores, entre otras, se encargará de calcular si el objetivo de la cámara del dispositivo apunta en la dirección a la que se encuentran los robots y la distancia a la que se encuentran para poder pintarlos. Es necesario por ello que el módulo de posicionamiento le facilite las coordenadas GPS tanto del usuario como de los robots a capturar.

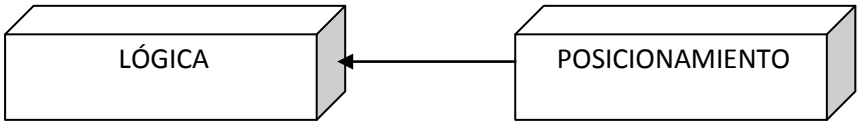
Interacción 3



El módulo de sensores, calculará la dirección a la que se encuentra el usuario de

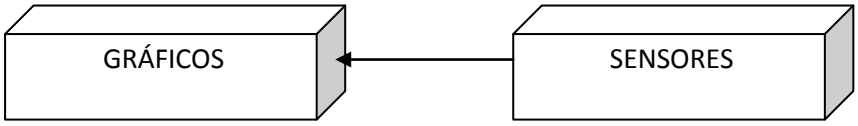
todos los robots que capturar en la ronda en la que está el videojuego, cuyas posiciones han sido generadas aleatoriamente por el módulo de lógica.

Interacción 4



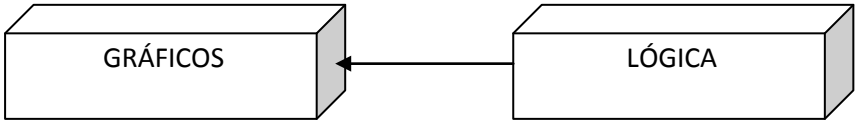
El módulo de lógica necesita de una serie de parámetros para desarrollar todas las funcionalidades que en este módulo se implementan. Como el posicionamiento de los robots y los ítems, o el desplazamiento de los robots. Para ello recibe del módulo de posicionamiento los atributos de la ubicación del usuario, el rango de coordenadas que se puede usar para colocar marcadores y área limitada de actuación.

Interacción 5



El módulo de gráficos se encargará de pintar los robots y los ítems que procedan en cada momento. Para saber tanto el tamaño como la posición a la que pintar éstos, necesita que el módulo de sensores le proporcione la distancia y el ángulo respecto al usuario en el que se encuentran. De la misma manera, para poder girar y rotar dichos objetos en función del dispositivo móvil, es necesario obtener los valores de pitch y roll, explicados en el siguiente capítulo, proporcionados por el módulo de sensores.

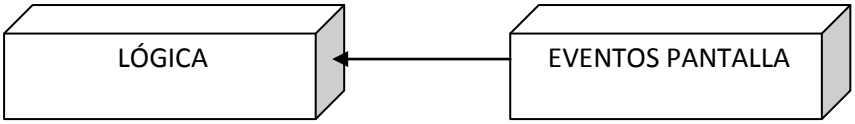
Interacción 6



Otro requisito para poder pintar tanto los robots como los ítems, es que éstos sean visibles desde la posición del usuario, es decir, que no haya ningún obstáculo entre

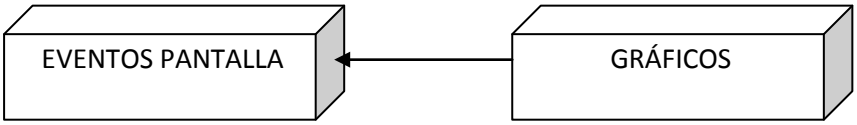
ambos que impida su visualización. Este dato es proporcionado por el módulo de lógica.

Interacción 7



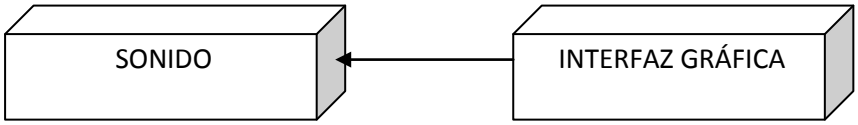
Para poder llevar a cabo las diferentes acciones que derivan de la captura de los robots o los ítems, tales como el paso a una ronda superior o disfrutar de las ayudas ofrecidas por los diferentes ítems, el módulo de lógica necesita que el módulo que se encarga de los eventos de la pantalla, le facilite cuándo éstos han sido capturados.

Interacción 8



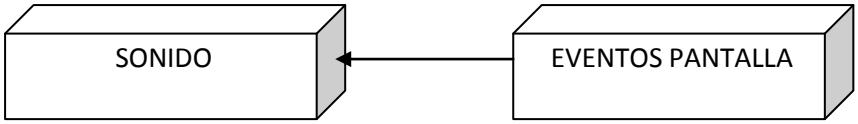
Para saber en qué posición de la pantalla están ubicados los robots o las ayudas a capturar, es necesario que el módulo de gráficos le facilite esta información al módulo de los eventos de pantalla.

Interacción 9



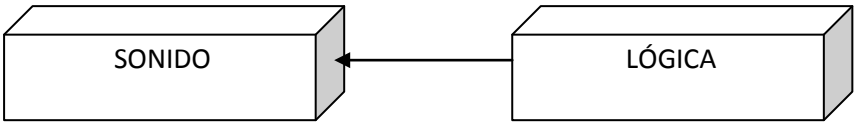
El módulo de sonido proporcionará efectos sonoros al videojuego siempre y cuando esta opción haya sido activada, o por lo menos no haya sido desactivada (ya que en caso de no elegir ninguna los sonidos se activan por defecto). Por ello necesita conocer el estado de activación o desactivación de los efectos, proporcionado por el módulo de interfaz gráfica.

Interacción 10



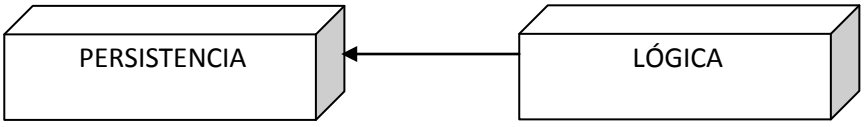
El módulo de sonido proporcionará efectos, siempre que éstos estén activados, cuando un robot o un ítem hayan sido capturados, por lo que necesitará esta información proporcionada por el módulo de eventos de pantalla.

Interacción 11



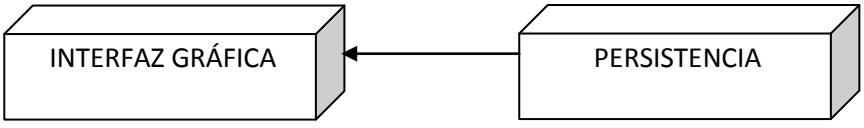
De la misma manera que en el caso anterior, el módulo de sonido proporcionará un efecto sonoro cuando el tiempo de la partida esté finalizando, información controlada por el módulo de lógica.

Interacción 12



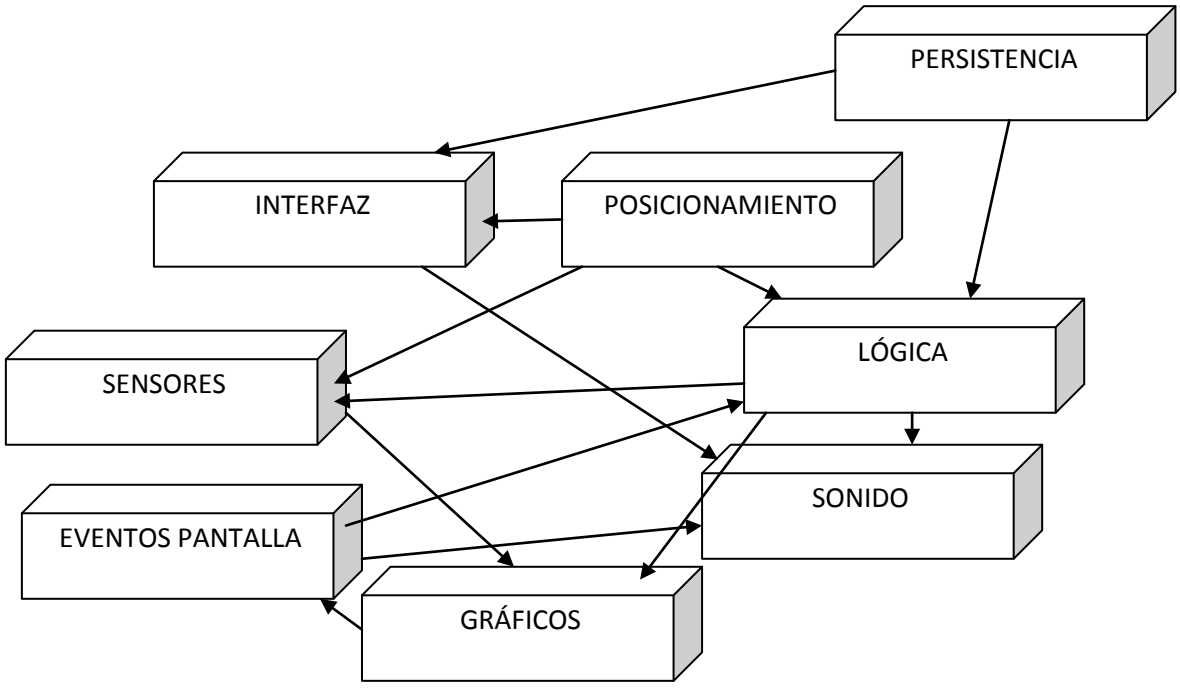
El módulo de lógica le facilitará al módulo de persistencia la puntuación obtenida por un usuario a lo largo de una partida. De esta manera se podrán almacenar los tres mejores resultados.

Interacción 13



En este caso será el módulo de persistencia el que ofrezca los resultados de las tres mejores puntuaciones a la interfaz gráfica, para que éste pueda mostrárselos al usuario, si es que desea visualizarlos.

Después de analizar las relaciones entre los diferentes módulos, quedaría el siguiente diagrama de interacción:



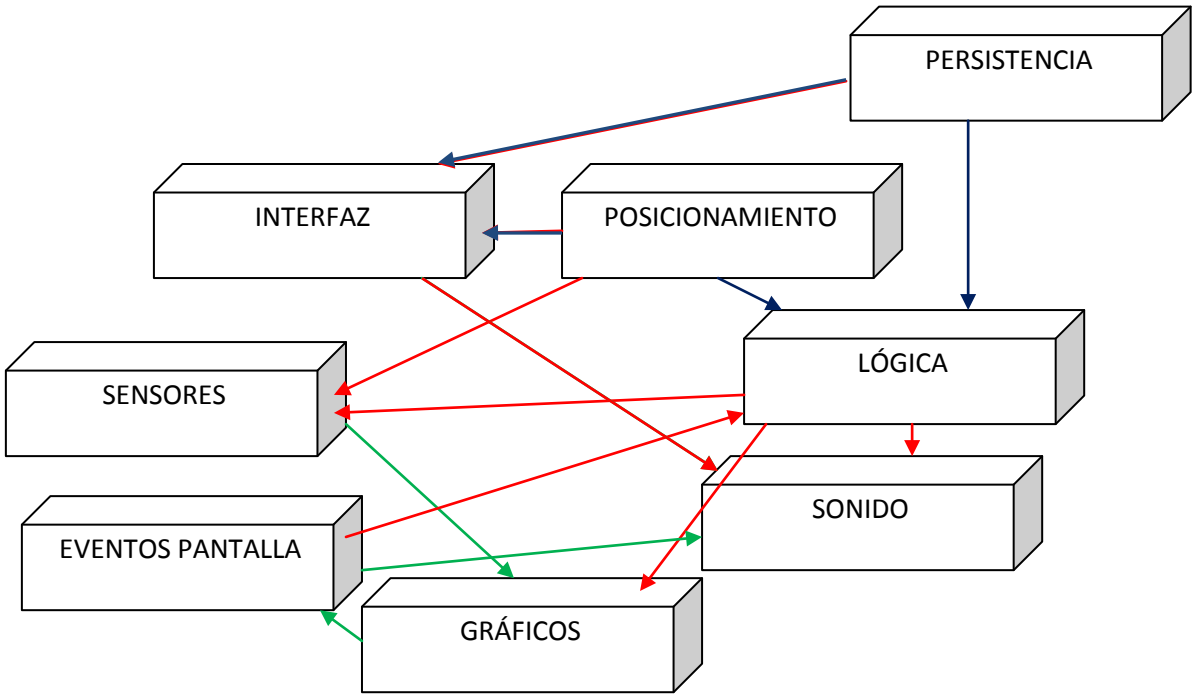
5.2.2. Descripción de los módulos

Puesto que el videojuego hace uso de la librería que se desarrolla en un proyecto previo, una vez analizado los módulos que lo componen, se deben identificarse cuáles se van a desarrollar y por tanto cuáles pertenecen a este proyecto, y cuáles se desarrollan como herramientas propias de la librería que se utiliza. Como se comentó en el capítulo anterior, este proyecto se encargaría de implementar la lógica del videojuego, de los aspectos relacionados con la localización del GPS y el API de Google Maps, y de la persistencia para almacenar las puntaciones más altas y poderlas mostrar de unas partidas a otras, además de crear la interfaz apropiada que lo conecte todo. Por lo tanto se deben implementar los módulos:

- Posicionamiento

- Lógica
- Persistencia
- Interfaz

Por tanto, podemos diseñar el diagrama de interacción, indicando cuáles de éstas se producen entre módulos del este proyecto, qué interacciones se llevan a cabo entre los módulos de este proyecto y los de la librería y por último las interacciones propias de la librería, lo que facilitará la integración del proyecto con la librería facilitada.



- Interacción entre los módulos pertenecientes a la librería del otro proyecto
- Interacción entre los módulos de este proyecto
- Interacción entre módulos del proyecto con la librería del otro proyecto

A continuación se describirán todos los módulos que componen el videojuego, haciendo más hincapié en los pertenecientes a este proyecto:

El módulo de Posicionamiento abarcará los aspectos de localización del usuario y la de los robots, teniendo en cuenta que el ámbito de trabajo queda reducido a las calles y que por tanto hay que controlar que las coordenadas de la situación del androide no correspondan con la situación de ningún edificio o con las de cualquier lugar inaccesible por el usuario con GPS. Además, será este módulo en el que se implemente el posicionamiento ficticio de los robots y la localización de los ítems, haciendo uso de las librerías de Google Maps.

El módulo de Lógica será el que comprenda todo lo relacionado con el funcionamiento del juego. Es decir, en este módulo se implementarán los movimientos de los androides, será el que controle si se ha capturado un robot, cada cuanto aparece un ítem, aplicar los beneficios de las ayudas o controlar la sucesión de las rondas. Además controlará la puntuación que vaya sumando el jugador a lo largo de la partida y el tiempo límite que tiene para terminar el juego.

El módulo de Interfaz de Usuario se encargará de, mediante sencillos menús navegables, manejar todos los aspectos configurables del juego a nivel usuario, así como permitir el acceso a información relativa al videojuego y sobre todo permitir el comienzo del mismo. Constará por tanto de una parte en la que se diseñará la interfaz gráfica y otra en la que se manejará todas las posibles opciones que se faciliten en ésta.

El módulo de sensores tendrá el principal cometido de pintar robots teniendo en cuenta que deberá pintarse sobre la imagen obtenida por la cámara del dispositivo del usuario atendiendo a la inclinación del terminal con respecto a la vertical y la horizontal de este, además de la distancia con el usuario.

El módulo de Gráficos será el encargado de generar todos los objetos de visualización que requieran el openGL, así como de su correcto funcionamiento. Así pues, será el encargado de integrar los gráficos de los robots y de los ítems.

El módulo de Sonido, se encargará de proporcionar los diferentes efectos sonoros cuando así se requieran.

El módulo de Eventos de pantalla, controlará las pulsaciones sobre la pantalla táctil del dispositivo, comprobando si se pulsa sobre un robot o un ítem que esté en condiciones de ser capturado.

El módulo de Persistencia será el encargado de almacenar las tres puntuaciones del juego más altas de todas las partidas jugadas a través de un fichero persistente relacionado al juego.

5.3. Implementación

En este punto se mostrará cómo se va a utilizar el análisis y diseño llevados a cabo en los apartados anteriores, implementando los módulos propuestos y teniendo en cuenta que la aplicación refleje las funciones presentadas en los Casos de Uso. Por otro lado, es necesario destacar que la finalidad del punto no es la de comentar el código de las clases, si no profundizar en todos los procesos que componen la fase de desarrollo. Por tanto, se especificará que sistema se ha utilizado para conseguir el posicionamiento del jugador y el mapa sobre el que se situarán tanto el jugador como los enemigos; cómo se han diseñado e implementando los enemigos y cuál es su comportamiento, se detallarán cómo se han situado sobre el mapa y cómo se ha creado la inteligencia artificial de estos y se especificará el diseño de las rondas y de todos los elementos relacionados con estos niveles. En definitiva cualquier aspecto relacionado con el desarrollo que se considere importante.

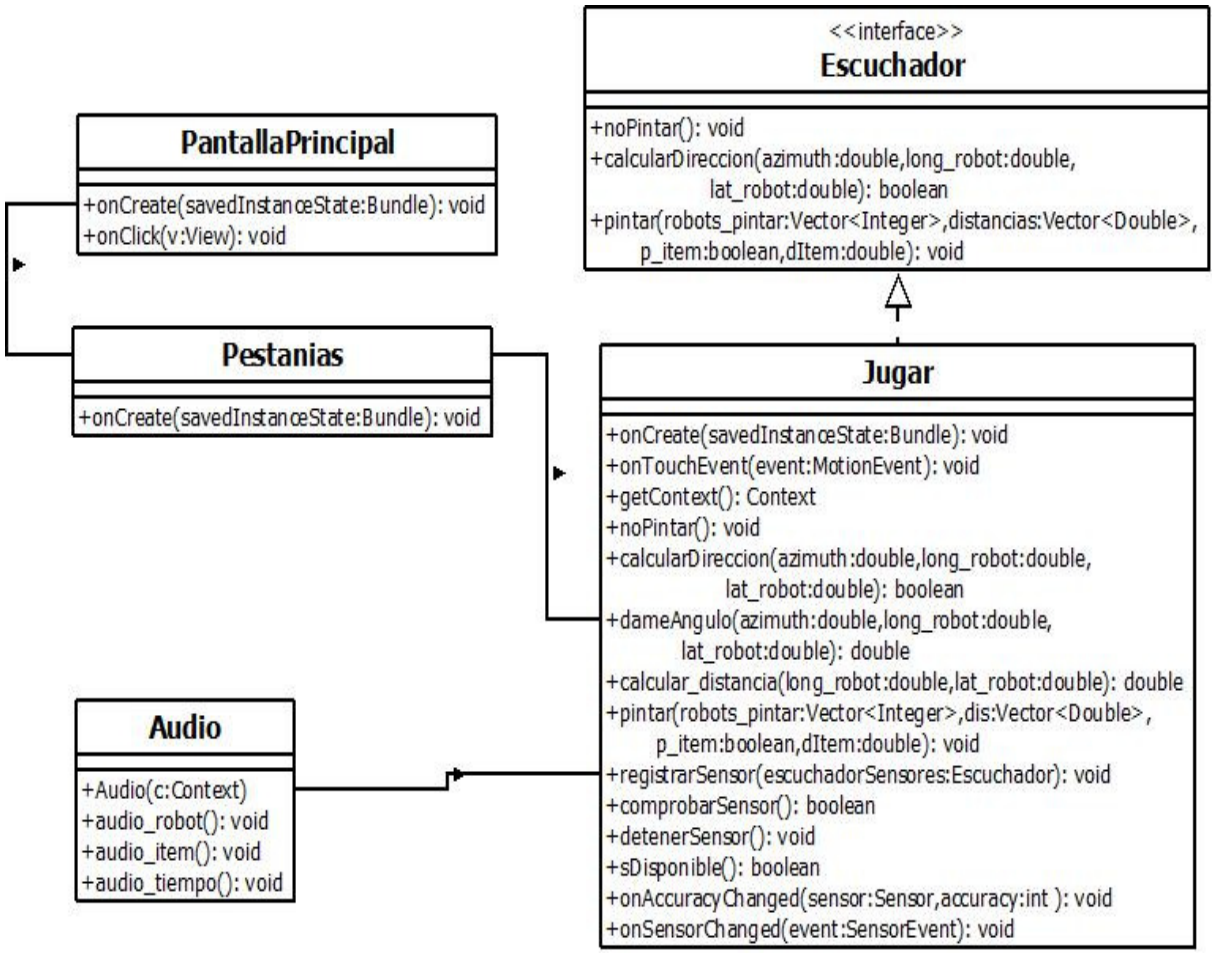
En cuanto a la organización del capítulo, será paralela al desarrollo real del juego, de forma que se comenzará por comentar aspectos de la implementación más relevantes y que constituirán la base del proyecto y se continuará explicando los diferentes elementos que se han implementado para completar las distintas funcionalidades del videojuego.

Pero antes de comenzar con todo la explicación del desarrollo del proyecto lo primero es mostrar un diagrama de clases donde toda la plenitud del proyecto quede expuesta. Además de antemano se pondrán de manifiesto las clases que están estrechamente relacionadas con las librerías del otro proyecto y con la implementación de esta mediante el contrato de interfaz.

5.3.1. Contrato de interfaz

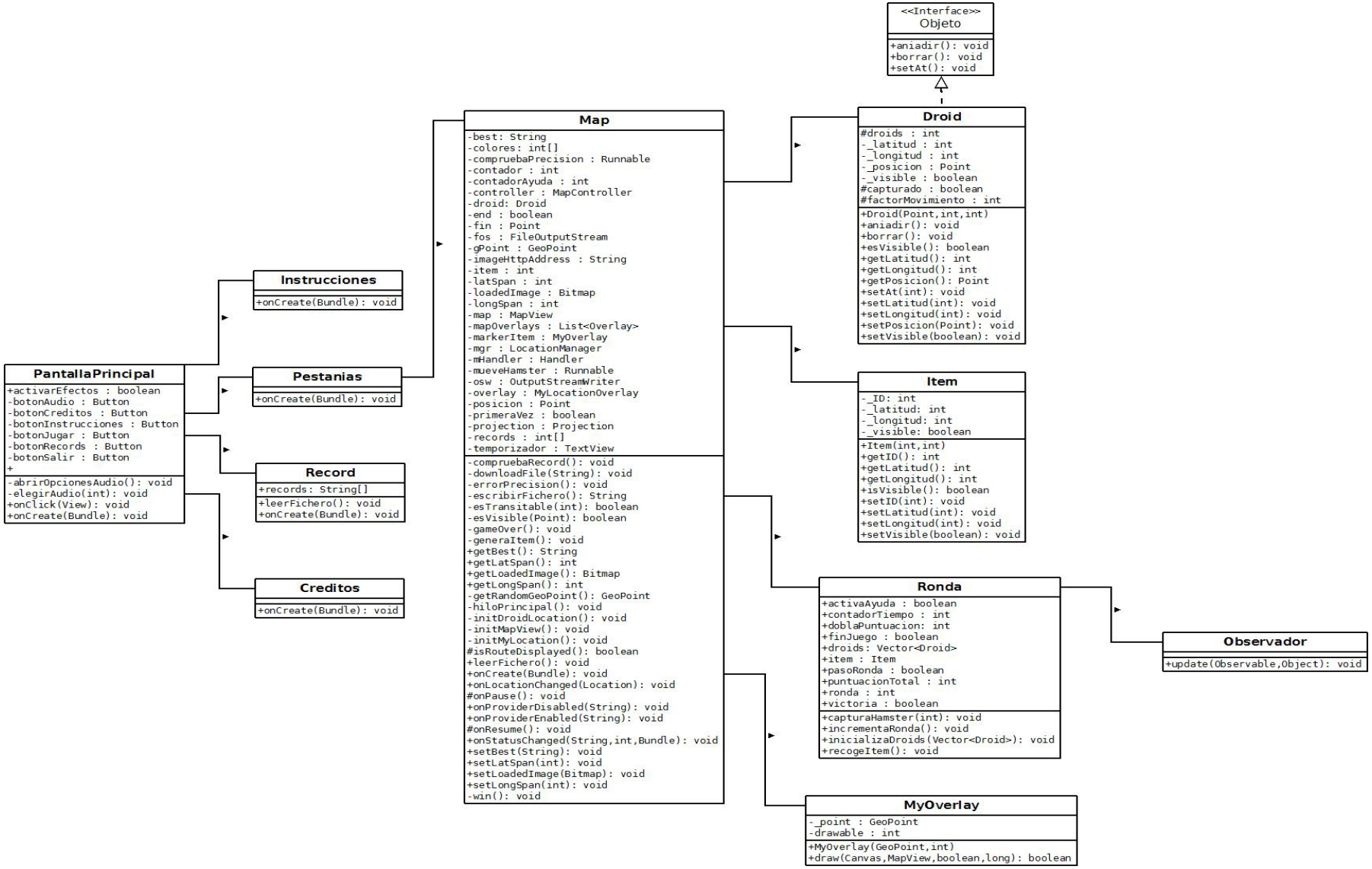
Para comenzar con el apartado de implementación en el que se explicara con detalle los pasos seguidos para la realización de este proyecto, es conveniente realizar un diagrama de clases de la parte que se va a integrar con el otro proyecto para poder completar el videojuego, es decir, las clases que participan en la interacción entre los dos proyectos y sus correspondientes métodos públicos.

Como se observa más adelante, el siguiente gráfico parte de dos clases propias de este proyecto, *PantallaPrincipal* y *Pestanias*, el resto, las clases que representan el uso de la librería desarrolladas en el otro trabajo sobre las que se basa éste.



5.3.2. Diagrama de clases

A continuación se muestran todas las clases que componen el proyecto que se ha desarrollado y que se expone en esta memoria.


<<Interface>>
Objeto
+aniadir(): void
+borrar(): void
+setAt(): void
Map
-best: String
-colores: int[]
-compruebaPrecision : Runnable
-contador : int
-contadorAyuda : int
-controller : MapController
-droid: Droid
-end : boolean
-fin : Point
-fos : FileOutputStream
-gPoint : GeoPoint
-imageHttpAddress : String
-item : int
-latSpan : int
-loadedImage : Bitmap
-longSpan : int
-map : MapView
-mapOverlays : List<Overlay>
-markerItem : MyOverlay
-mgr : LocationManager
-mHandler : Handler
-mueveHamster : Runnable
-osw : OutputStreamWriter
-overlay : MyLocationOverlay
-posicion : Point
-primeraVez : boolean
-projection : Projection
-records : int[]
-temporizador : TextView
-compruebaRecord(): void
-downloadFile(String): void
-errorPrecision(): void
-escribirFichero(): String
-esTransitable(int): boolean
-esVisible(Point): boolean
-gameOver(): void
-generaItem(): void
+getBest(): String
+getLatSpan(): int
+getLoadedImage(): Bitmap
+getLongSpan(): int
-getRandomGeoPoint(): GeoPoint
-hiloPrincipal(): void
-initDroidLocation(): void
-initMapView(): void
-initMyLocation(): void
#isRouteDisplayed(): boolean
+leerFichero(): void
+onCreate(Bundle): void
+onLocationChanged(Location): void
#onPause(): void
+onProviderDisabled(String): void
+onProviderEnabled(String): void
#onResume(): void
+onStatusChanged(String,int,Bundle): void
+setBest(String): void
+setLatSpan(int): void
+setLoadedImage(Bitmap): void
+setLongSpan(int): void
-win(): void
Droid
#droids : int
-_latitud : int
-_longitud : int
-_posicion : Point
-_visible : boolean
#capturado : boolean
#factorMovimiento : int
+Droid(Point,int,int)
+aniadir(): void
+borrar(): void
+esVisible(): boolean
+getLatitud(): int
+getLongitud(): int
+getPosicion(): Point
+setAt(int): void
+setLatitud(int): void
+setLongitud(int): void
+setPosicion(Point): void
+setVisible(boolean): void
Item
-_ID: int
-_latitud: int
-_longitud: int
-_visible: boolean
+Item(int,int)
+getID(): int
+getLatitud(): int
+getLongitud(): int
+isVisible(): boolean
+setID(int): void
+setLatitud(int): void
+setLongitud(int): void
+setVisible(boolean): void
Ronda
+activaAyuda : boolean
+contadorTiempo : int
+doblaPuntuacion: int
+finJuego : boolean
+droids: Vector<Droid>
+item : Item
+pasoRonda : boolean
+puntuacionTotal : int
+ronda : int
+victoria : boolean
+capturaHamster(int): void
+incrementaRonda(): void
+inicializaDroids(Vector<Droid>): void
+recogeItem(): void
Observador
+update(Observable,Object): void
MyOverlay
-_point : GeoPoint
-drawable : int
+MyOverlay(GeoPoint,int)
+draw(Canvas,MapView,boolean,long): boolean
Instrucciones
+onCreate(Bundle): void
Pestanias
+onCreate(Bundle): void
Record
+records: String[]
+leerFichero(): void
+onCreate(Bundle): void
Creditos
+onCreate(Bundle): void
PantallaPrincipal
+activarEfectos : boolean
-botonAudio : Button
-botonCreditos : Button
-botonInstrucciones : Button
-botonJugar : Button
-botonRecords : Button
-botonSalir : Button
+
-abrirOpcionesAudio(): void
-elegirAudio(int): void
+onClick(View): void
+onCreate(Bundle): void

5.3.3. Detalles de la implementación

A lo largo de este apartado se explicará el procedimiento seguido para el desarrollo del proyecto describiendo de manera que se han ido implementado las diferentes funcionalidades para conseguir los objetivos marcados. De esta manera se vuelve bastante importante que, en un principio, se centre este apartado en la explicación de cómo conseguir la vista de un mapa a partir del API de Google Maps y como centrarlo en la posición, obtenida por GPS, de nuestro usuario. Pero antes se explicará cómo se ha creado la interfaz de usuario. A continuación la implementación continúa con la adhesión de diferentes funcionalidades que se vuelven imprescindibles a la hora de implementar la lógica del juego.

Para hacer más sencillo el entendimiento de la implementación este apartado se dividirá en diferentes sub-apartados en los que se abordarán el desarrollo de las diferentes funcionalidades. Así se conseguirá una explicación más fluida que permitirá ahondar en cada una de las fases del desarrollo.

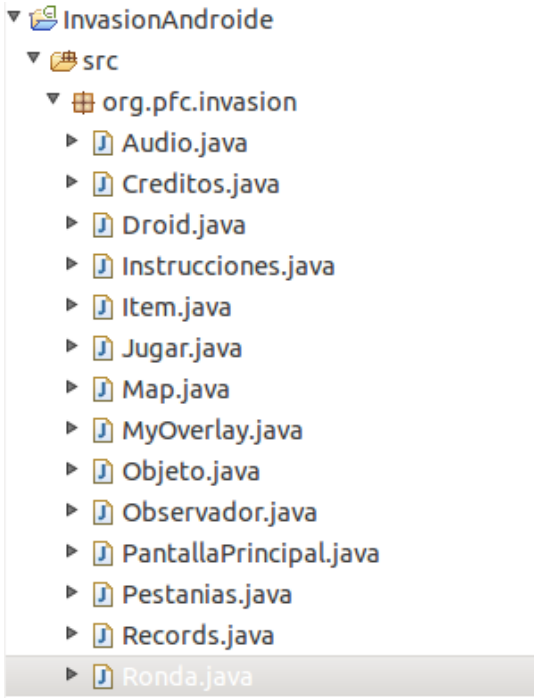
Interfaz gráfica

El primer paso en la realización de un videojuego es el diseño de la interfaz gráfica la cual nos permite, en primer lugar, proporcionar al usuario un entorno visual sencillo para poder utilizar todas las funcionalidades del videojuego, y en segundo lugar familiarizarnos, en este caso, con el entorno de programación Android.

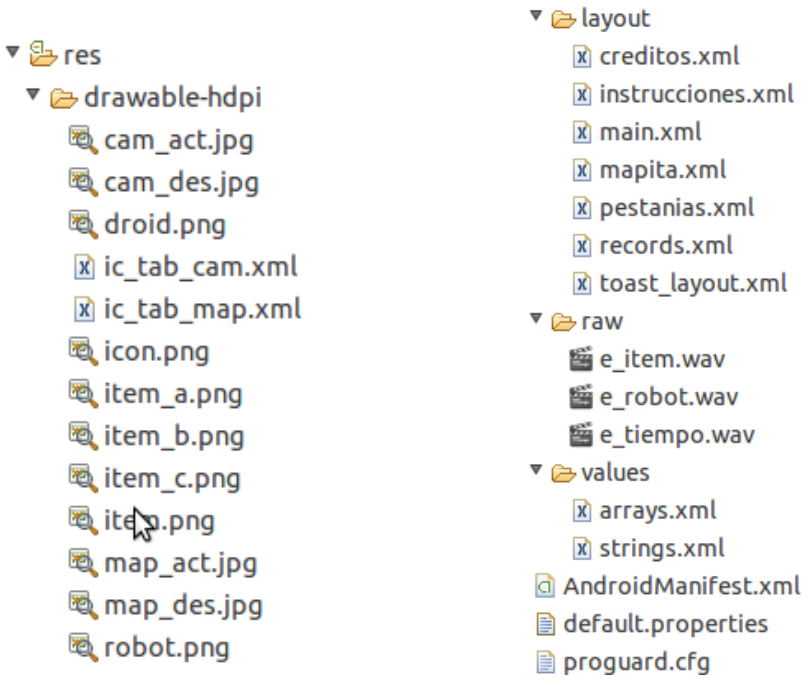
Una curiosidad del sistema Android es que nos ofrece la posibilidad de crear las interfaces graficas a través ficheros XML, lo que nos permite separar la funcionalidad de las clases y la estética. De esta forma, nuestro código queda mucho más ordenado y claro, indicando en los archivos XML la distribución de nuestros objetos gráficos (botones, pestanas…), así como su color, forma, tamaño, colocación etc., mientras que en los archivos java podemos centrarnos en la funcionalidad de estos y su integración en el sistema.

Veamos primero como se organizan todos estos ficheros y archivos en el Eclipse para poder crear un proyecto Android.

El proyecto consta de un único paquete, org.pfc.invasion, en el que se encuentran todas las clases.



Dentro del proyecto, hay varias carpetas. En las carpetas drawable, guardamos todas las fotos que vamos a utilizar; en layout, los ficheros xml correspondientes a la interfaz de usuario; en raw, los archivos de música que se usaran para los efectos de sonido; y en values, los ficheros xml donde guardaremos el valor de algunos strings que se utilizan en el proyecto.



Por último se encuentra el fichero Manifiesto, archivo xml donde se reflejan los componentes de la aplicación y la relación entre ellos, las bibliotecas necesarias (además de las de Android) y los permisos requeridos.

A continuación se presentan las clases pertenecientes a la interfaz grafica:

Clase PantallaPrincipal

Esta clase extiende de la clase Activity, la cual permite interactuar con el usuario mostrando la información correspondiente por la pantalla. Por ello aquí definiremos los botones del menú principal en un archivo XML, y la funcionalidad de éstos, en el documento java. La pantalla principal consta de 6 botones: Jugar, Instrucciones, Opciones de Audio, Records, Créditos y Salir.

- Jugar: desde el botón jugar, comienza la partida del videojuego. Se abre una pantalla con 2 pestañas, mostrando en la de por defecto el mapa de Google con la posición del usuario y los robots a encontrar, y en la otra pestaña la imagen de la cámara.
- Instrucciones: muestra al usuario las instrucciones del juego.
- Opciones de Audio: abre un ALertDialog que permite al usuario activar o desactivar los efectos de sonido.
- Récords: muestra en la pantalla las 3 puntuaciones más altas logradas en el videojuego.
- Créditos: muestra al usuario los créditos del videojuego.
- Salir: cierra la aplicación.

Implementa la interfaz OnClickListener, para manejar los eventos táctiles sobre los botones, por lo que asignamos los botones al escuchador y las funcionalidades correspondientes a cada botón.

```
@Override
public void onClick(View v) {
        switch (v.getId()){
                case R.id.bJugar:
                        Intent i1 = new Intent (this, TabHost.class);
                        startActivity(i1);
                break;

                case R.id.bInstrucciones:
                        Intent i2 = new Intent (this, Instrucciones.class);
                        startActivity(i2);
                break;

                case R.id.bAudio:
                        abrirOpcionesAudio();
                break;

                case R.id.bRecords:
                        Intent i3 = new Intent (this, Records.class);
                        startActivity(i3);
                break;

                case R.id.bCreditos:
                        Intent i4 = new Intent (this, Creditos.class);
                        startActivity(i4);
                break;

                case R.id.bSalir:
                finish();
                break;
        }
}
```

Para activar o desactivar los efectos de audio, creamos una variable estática de tipo booleano, y mediante un *AlertDialog*, mostrado al usuario al abrir las opciones de audio, le asignamos a la variable el valor true si se elige la opción de activar o le asignamos false, si se elige la opción de desactivar. Por defecto la variable toma el valor true, ya que si no se indica lo contrario los efectos de sonido estarán activados.

Clase Pestanias

Extiende de la clase TabActivity, y es la encargada de mostrar al usuario las dos pestañas disponibles a elegir para la visualización del juego: el mapa donde se muestra la localización de los robots, o la imagen de la cámara del dispositivo.

Clase Instrucciones y Clase Créditos

Clases que muestran en una pantalla sobre el menú principal, las instrucciones y los créditos del videojuego, respectivamente. Añadimos un tema en el manifiesto para dar mayor estética a la visualización de dichas pantallas.

Clase Records

La clase Records muestra un ListView en el que se insertan las 3 mejores puntuaciones logradas en todas las ejecuciones del juego.

MapView

Se hace necesario continuar profundizando en el proceso de desarrollo del eje central de este proyecto, la vista del mapa que se centrará en la localización del usuario que inicie la partida en un momento determinado. Esta vista es importante porque se hace imprescindible para el resto de funcionalidades que implementan el videojuego, o al menos con las que interactúa el usuario, como los androides o los ítems.

La vista del mapa que se descargará constituye el tablero en el que el usuario se moverá y en el que se posicionarán los enemigos y las ayudas del jugador. Hay es dónde radica la importancia del mapa y lo que hace que este se convierta en la parte principal de este proyecto.

Aquí surge uno de los primeros interrogantes a la hora de seleccionar el tipo de mapa que iremos a utilizar como punto de inicio del videojuego, ya que como se ha podido ver hay diferentes opciones.

Las dos opciones más evidentes que se presentaban eran o bien usar un mapa estático o bien uno dinámico. Las ventajas que tendríamos con el estático es que se podría acceder al color de cada pixel, algo que como veremos terminará siendo muy importante, además de tener un área fija con un zoom determinado del que los androides no pudieran moverse. Por el contrario el mapa dinámico nos facilitaba un método que nos permitía tener una correlación entre coordenadas y pixeles, algo realmente necesario, además de marcarnos automáticamente la posición del usuario y el radio de error de GPS sobre el mapa.

Debido a que las ventajas del mapa estático no eran tan significativas como las que aportaba el dinámico, ya que se le pueden quitar los controles del zoom y el desplazamiento a lo largo de la pantalla, se decidió utilizar un mapa dinámico. No obstante la obtención del color de los pixeles será un punto muy necesario, por lo que no se desecho del todo esta opción como se puede ver más adelante.

Antes de comenzar a implementar ninguna aplicación se hace necesario el tener un entorno de desarrollo adecuado, es por esto que las primeras líneas de este sub-apartado se dedican a explicar que elementos son necesarios para crear este entorno.

Se impone como requisito tener la biblioteca externa de Google Maps instalada en el entorno de SDK. La biblioteca de mapas se incluye en el API de Google, y se puede instalar con el SDK de Android y AVD Manager. Los pasos para instalarlos se

pueden encontrar fácilmente en la página oficial para desarrolladores de android, accediendo a la página y consultándolo se puede aprender cómo hacerlo y cómo utilizarlo en beneficio de nuestras aplicaciones.

Después de instalar el API de Google, se tendrá que seleccionar, a la hora de crear nuestra aplicación el target llamado "Google APIs by Google Inc.". También se tendrá que crear una nueva AVD que utilice el mismo destino de despliegue de las API de Google.

Además para poder utilizar la API de Google Maps se requiere la obtención previa de una clave de uso [Anexo D. Obtención API Key], que estará asociada al certificado con el que firmamos digitalmente nuestra aplicación. Esto quiere decir que si se cambia el certificado con el que se firman las aplicaciones (algo que se hace normalmente como paso previo a la publicación de la aplicación en el Market) se tendrá que cambiar también la clave de uso de la API.

Una vez contamos con la clave de uso de la API, la inclusión de mapas en nuestra aplicación es una tarea relativamente sencilla y directa.

Con estos pasos ya habremos creado el entorno de desarrollo en Eclipse para la creación de la actividad principal de nuestro videojuego la cual tendrá un mapa de Google como vista principal.

Ahora bien, para introducir esta vista en nuestra actividad hay que seguir una serie de puntos.

Lo primero es iniciar un nuevo proyecto que extienda de la clase *MapActivity*, para implementar todas las funcionalidades que queremos que nuestra aplicación contenga, aparte de la que centra este sub-apartado.

Surge un problema, y es que, debido a que la biblioteca de mapas no es una parte de la biblioteca estándar de Android, se debe declarar en el Manifiesto de Android. Para ello se abrirá el archivo manifest.xml y se agregará como hijo de la etiqueta <application> el siguiente elemento:

```
<uses-library android:name="com.google.android.maps" />
```

También se hace necesario el acceso a Internet, con el fin de recuperar los mapas, por lo que también se debe solicitar el permiso INTERNET. Nuevamente en el archivo de manifiesto se deberá agrega la siguiente etiqueta como hijo del elemento <manifest>:

```
<uses-permission android:name="android.permission.INTERNET/>
```

A continuación, para conseguir insertar el mapa que descargamos de internet, se tiene que abrir el archivo res/layout/main.xml y añadir una etiqueta al nodo raíz com.google.android.maps.MapView que haga de contenedor:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
       android:orientation="vertical"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent">

<com.google.android.maps.MapView
       android:id="@+id/map"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       android:apiKey="0L6GThKD5rfxEhrUMZHa1l3AD…"
       android:clickable="false" />
</LinearLayout>
```

El contenedor elegido se trata de un `LinearLayout`, el cual tiene una serie de parámetros que a continuación se explican. El atributo android:clickable define si se desea permitir la interacción del usuario con el mapa. Si este es "false" y luego el usuario intenta tocar el mapa no hace nada, en nuestro caso tomará este valor para así mantener inalterado el mapa sobre el que se basará el resto del trabajo. El atributo android:apiKey es la clave API de Google Maps que se tiene que obtener. Esta clave es necesaria para recibir los datos del mapa, incluso mientras se está desarrollando.

Ahora, como ya se había indicado la clase que se implementa tiene que extender de MapActivity, en lugar de la actividad que se acostumbra a extender de android.app.Activity, como podemos ver a continuación:

```
public class Map extends MapActivity{

            .
            .
            .
}
```

Esta es una sub-clase de Activity, facilitada por la biblioteca Maps, que ofrece importantes capacidades al mapa, necesarios para el proyecto.

Al heredar nuestra clase de MapActivity se debe implementar obligatoriamente el método isRouteDisplayed(), cuyo valor de retorno debe ser true sólo en caso de que se vaya a representar algún tipo de información de ruta sobre el mapa (esto no se trata de ningún tema técnico, sino tan sólo legal, para cumplir los términos de uso de la API de Google Maps). Tendremos que sobrescribir este método y, para el caso que se implemente, devolver el valor de retorno *false*, como se muestra:

```
@Override
protected boolean isRouteDisplayed(){
     return false;
}
```

Además de éste, en el método onCreate(), para que implemente algunas funcionalidades extras se tiene que llamar al método estándar onCreate() de la clase que se hereda, como se puede observar aquí.

```
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
}
```

Esto carga el archivo creado anteriormente. De hecho, esta aplicación está completamente funcional y mostrará las baldosas de cualquier mapa. Pero no existe ninguna capacidad de zoom. Si se desea añadirla se llamará al método setBuiltInZoomControls() sobre la referencia al control MapView para mostrar los controles de zoom estándar sobre el mapa, de forma que podamos acercar y alejar la vista del mapa. En el caso de este videojuego se pondrán los controles de zoom a false para que no se muestren y no se puedan variar el tamaño mostrado.

Hay que añadir que la invocación de métodos para deshabilitar los controles en el videojuego que se está desarrollando se ha introducido en otro método denominado initMapView(), y que se invoca en el método onCreate(), a diferencia de como se ha explicado antes. Este método se muestra a continuación.

```
private void initMapView() {
    map = (MapView)findViewById(R.id.map);
    controller = map.getController();
    map.setBuiltInZoomControls(false);
}
```

Con esto, ya tendríamos un mapa visible en nuestra aplicación que cumpla con todos los requisitos que deseamos tener disponibles.

Localización

A continuación se hablará del procedimiento que se sigue para conseguir la ubicación del usuario y la manera que si utiliza esta localización para centrar la imagen del mapa.

Se tiene que tener en cuenta que la situación del usuario ha de estar disponible en todo momento, es por esto que se introduce la solicitud de la posición en un hilo que se estará ejecutando paralelamente a otros procesos en curso dentro del juego.

El API de Google Maps nos proporciona los métodos necesarios para introducir un marcador con la localización del jugador, por lo que solo tenemos que implementarlos para tener esta posición. Toda esta funcionalidad se implementa en el método `initMyLocation()`que se muestra a continuación.

```
private void initMyLocation() {
```

```
        overlay = new MyLocationOverlay(this, map);
        overlay.enableMyLocation();
        overlay.runOnFirstFix(new Runnable(){
            public void run(){
                controller.setZoom(18);
                controller.animateTo(overlay.getMyLocation());
            }
        });
        map.getOverlays().add(overlay);
    }
```

Como podemos ver se hace uso del método `enableMyLocation()` de la clase `MyLocationOverlay` para activar la localización. Pero no es hasta que se hace la invocación del método `getMyLocation()` cuando se obtiene la ubicación exacta del usuario. En esta misma línea se hace uso del método `animateTo()` para centrar la imagen del mapa en esta localización. Al final del método se añade el marcador del usuario a la vista del mapa que tiene disponible.

Precisión GPS

Una vez que se tiene la vista del mapa disponible, ya se dispone de la base funcional del videojuego. Pero no obstante, antes de empezar a desarrollar la lógica del juego, se tiene que atender una funcionalidad muy importante, asegurar una mínima precisión GPS. Ya que si ésta no se cumple no se podrá comenzar el juego ni mantener la jugabilidad con fluidez.

Para ello nuestra clase tiene que implementar LocationListener, para poder escuchar todos los eventos del sensor GPS de nuestro terminal. Ésta a su vez obliga a sobrescribir una serie de métodos propios de esta clase para hacer uso de sus funcionalidades. Estas clases son onLocationChanged, onProviderDisabled, onProviderEnabled y onStatusChanged. Con estos pasos ya se puede desarrollar un control de precisión para el juego.

Una vez inicializadas la vista del mapa que se muestra al jugador, lo que se hará es, a través de un manejador de localizaciones, clase que implementa Android bajo el nombre de LocationManager, obtener nuestra posición GPS de la clase Location, la cual permitirá llamar al método getAccuracy(), que nos facilitara la precisión de esa localización.

Pero para obtener esta localización se necesita un proveedor, el cual se obtiene con el mismo manejador. Y que, como es evidente queremos que sea el mejor. Por ello hacemos uso de la clase Criteria, que nos permite hacer una búsqueda en función de un criterio de búsqueda que marcará un patrón. Este patrón será que de la precisión más próxima, lo que indicaremos de la siguiente manera Criteria.ACCURACY_FINE.

Con este patrón de búsqueda se puede invocar al método getBestProvider() de la clase LocationManager para que nos dé el mejor proveedor para nuestra localización.

Todo esto podemos verlo ejemplificado a continuación.

```
mgr = (LocationManager) getSystemService(LOCATION_SERVICE);
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_FINE);
best = mgr.getBestProvider(criteria, false);
Location location = mgr.getLastKnownLocation(best);
```

Una vez inicializa nuestra posición podemos llevar un control de la precisión que el proveedor nos facilita. Ahora bien que el control que se realice tendrá que ser rutinario, es por esto que el control de esta rutina se hará a través de un hilo. Esto permitirá que si la precisión se pierde en algún momento tenerlo controlado.

Androides

Todo videojuego tiene un objetivo, éste, como ya sabemos, se centra en eliminar a todos los androides que en una ronda aparecen, para ello hay que capturarlos desplazándonos a una posición próxima a la que este se encuentra.

Tenemos que tener en cuenta distintas cosas en relación al diseño de los androides:

- Estos estarán, caracterizados por unas coordenadas en el mundo real, que nos servirán, por ejemplo, para calcular distancias. De hecho, estás serán uno de los puntos más importantes de unión entre los dos proyectos, siendo el punto de partida junto a las coordenadas que ocupa el usuario.

- Estarán también caracterizados por un punto, un coordenada x y una coordenada y, que posicionarán de manera inequívoca los androides en el mapa de bits obtenido como la proyección de la vista del mapa que nos facilita Google Maps.

- Hay que tener en cuenta que estos androides se desplazan, limitados como veremos más adelante por la primera proyección obtenida del mapa, pero se mueven. Por lo que habrá que crear una rutina que permita a estos desplazarse acordes con su inteligencia artificial.

- Además, puesto que el desarrollo del juego sucede por la ciudad, hay que tener en cuenta que los puntos por los que los androides se pueden desplazar tienen que ser transitables por cualquier persona, por lo que tenemos que diseñar algún mecanismo que nos permita ubicarlo, en un principio, en una posición adecuada y más tarde evitar que se desplace

hacia zonas no transitables.

- Por último, debemos idear una funcionalidad que nos permita determinar si, desde la posición que nos encontramos y siempre que dirigiéramos la mirada hacia el lugar idóneo, podría verse el androide al que perseguimos para capturar. Esta funcionalidad es necesaria implementarla como otro de los aspectos de comunicación entre los dos proyectos, ya que en el proyecto de mi compañera solo se podrá divisar un androide si la cámara apunta a la dirección adecuada y no hay ningún obstáculo que impida verlo, siempre que se encuentren dentro de un rango menor de 20 metros. Con obstáculo nos referimos a algún edificio que obstruya la visualización directa.

Prácticamente son estos puntos los que se tuvo que salvar para crear unos enemigos autómatas, mínimamente aceptables, que se convirtieran en el objetivo de este videojuego.

Para tener caracterizados a los androides y sus diferentes funcionalidades se diseñó una clase que cumpliera los requisitos mininos expuestos antes, *Droid*. Esta clase permite instanciar a los enemigos e identificarlos con un ID, unas coordenadas y un punto de referencia sobre la proyección del mapa. Además consta de un atributo que se actualiza, en función de un método que explicaremos más adelante, y determina en cada momento si un androide puede ser divisado por un jugador.

A la hora de explicar la implementación de las funcionalidades de un robot, se van a crear unos puntos en los que se afronten cada una de estas funcionalidades, así se facilita su comprensión.

1. Posicionamiento Androides

Es necesario generar unas coordenadas en las que posicionar a los diferentes robots de un nivel, unas coordenadas que serán aleatorias. De esta manera se consigue que cada partida sea distinta, ya que rara vez sus posiciones coincidirán.

Cada vez que una partida se inicia se generan aleatoriamente la posición de los androides de la primera ronda teniendo en cuenta que tienen que estar en un radio de distancia límite para que el usuario pueda capturarlos y que deben situarse sobre una superficie transitable.

Para solventar el posicionamiento aleatorio de los androides lo que se hizo fue crear un método `getRandomGeoPoint()`, que devolviera un punto geográfico aleatorio, utilizando el método `Math.random()`, caracterizado por la latitud y la longitud, que se encontrara dentro de un área que a continuación veremos cómo se definió.

El API de Google Maps tiene un par de métodos, de la clase MapView, que te permiten conocer la diferencia que existe entre las coordenadas que caracterizan al pixel de la esquina superior izquierda y al pixel situado en la esquina superior derecha de la pantalla del terminal que se está utilizando. De igual manera te permite conocer la diferencia entre el pixel de la esquina superior izquierda y el pixel de la esquina inferior izquierda. Estos métodos son `getLatitudeSpan()` y `getLongitudeSpan()` respectivamente.

Esto nos permite, con las coordenadas de un punto dentro del mapa de la pantalla, tener caracterizados todos los puntos de la pantalla con sus respectivas coordenadas geográficas. Como conocemos la posición del usuario, que adrede hemos centrado en el mapa que mostramos en la pantalla de nuestro dispositivo, tenemos todos los puntos caracterizados.

Ahora bien, estos métodos solo se pueden usar en el momento que todas las funcionalidades del mapa estén estabilizadas, es decir, que la visión del mapa esté cargada correctamente y que el GPS nos dé una señal lo suficientemente precisa para que no sufra alteraciones al centrar el mapa en la posición del jugador. Para cerciorarse que todas las funcionalidades del mapa son estables, se decide hacer uso de estos métodos en el método `onResume()`, que toda Actividad en Android tiene que tener.

En cambio para generar el posicionamiento aleatorio de los enemigos además de tener que usar estos métodos es necesario tener la suficiente precisión de GPS. Es por esto que a continuación de haber detectado una precisión mínima aceptable, en el hilo que controla la rutina de la precisión, se invoca el método `initDroidLocation()`, que será el encargado de generar tantas instancias de la clase `Droid`, con sus respectivas posiciones geográficas aleatorias, como corresponda en la ronda.

Llegados a este punto, sólo queda explicar cómo se obtienen las coordenadas aleatorias de cada androide. Para hacerlo más sencillo se centrara la explicación en cómo se obtiene, por ejemplo, la latitud.

El primer paso es obtener la diferencia de distancia de latitudes dentro de la pantalla, esta viene dada por el método `getLatitudeSpan()`, valor que se almacena en la variable `latSpan`. A continuación se deberá obtener una latitud aleatoria entre cero y el valor dado por la variable `latSpan`. Una vez que se consigue este valor lo único que se tiene que hacer es sumárselo a la latitud menor que se puede encontrar en nuestra pantalla. Para hallar este valor lo que hay que hacer es coger la latitud del par de coordenadas que caracterizan la localización del jugador, que hasta el momento coinciden con el centro de la pantalla, y se le resta la mitad de la diferencia de latitudes de nuestra pantalla, reflejada en la variable `latSpan`. Todo este procedimiento se resume en una línea de código:

```
randomLat = Math.random()*latSpan
+(overlay.getMyLocation().getLatitudeE6()-latSpan /2));
```

De esta manera se puede concluir la obtención de la primera coordenada de la localización de un androide. Para la otra, la longitud, el procedimiento es exactamente el mismo, solo que cambiando el método getLatitudeSpan por el método getLongitudeSpan y la variable latSpan por la variable longSpan.

2. Accesibilidad

Este fue uno de los aspectos más complicados de salvar a la hora de realizar el proyecto, al que se dedicó mucho más tiempo del proyecto por una serie de problemas. Este problema se resolvió mediante un procedimiento de prueba y error, en el que se planteaba una forma de solucionarlo, se desarrollaba y al no obtener los resultados deseados se desechaba y realizaba otra. Y con todo la solución tomada no consigue unos resultados exacto, aunque sí bastante validos para el objetivo de este proyecto.

Comenzamos desarrollando una idea en la que se hacía uso de una tercera coordenada para determinar si el posicionamiento de los androides era válido. Pero los mapas de Google en 3D no están desplegados por cualquier área y se desconocen las latitudes de muchos puntos, por lo que se podían colocar objetivos en puntos que no fueran accesibles.

La siguiente solución que se implementó fue la de tomar como accesibles únicamente aquellos pixeles que correspondieran al color blanco de las calles. Pero necesitábamos una correlación entre coordenadas y color del pixel que esas coordenadas correspondían, esto suscitó el cambio de usar como referente un mapa dinámico, pero éste a su vez necesitaba una imagen o un mapa estático del que obtener el color. Para ello teníamos que usar la combinación del mapa dinámico, para determinar en qué pixel se estaba situando el enemigo en función de las coordenadas, y el estático para conocer el color de dicho pixel. No obstante, por razones que no se han logrado determinar, esta solución forzaba el cierre de la aplicación. Posiblemente fue debido a que el color buscado, en relación con el resto, era muy difícil de encontrar, y al buscar el color, tras un rato, la aplicación forzaba su cierre.

Por el contrario se uso un vector de colores en los que no se podría situar un androide, esto facilitaba mucho el encontrar un punto en el que situarlo y no forzaba el cierre de la aplicación. Pero en ocasiones la posición no era accesible, por lo que una vez determinado este punto, se comprobaba que se trataba de un punto de color de calles transitables o de parques. A continuación se expone con detalle todo el procedimiento seguido.

Para superar con éxito el segundo problema que se plantea a la hora de posicionar los enemigos del juego, que se sitúen en una posición accesible, se ha de comprobar que el par de coordenadas generadas aleatoriamente corresponden con una zona transitable. Si no se tratara de una zona transitable se tendría que generar de nuevo un par de coordenadas nuevas. Es por esto por lo que el paso anterior ha de repetirse mediante un bucle hasta que se cumpla la condición de que sea transitable. El bucle

elegido será un bucle *while* cuya condición para repetir el proceso anterior será una variable booleana, que indique si es transitable, denominada *transitable*, definiéndose el bucle así:

```
while(!transitable){

        .
        .
        .

}
```

Dentro del bucle es donde se evaluará el punto geográfico que se genera y donde la variable `transitable` se verá alterada en función de esta evaluación.

Para evaluar el punto geográfico hacemos uso de un método que llamamos esTransitable(), al que pasaremos como parámetro a evaluar un color definido por un entero. Este color pertenece al pixel que caracteriza la localización de un androide, que es eventual, hasta que se demuestre que es transitable. Por tanto para la obtención de esta variable se ha de llevar un orden, primero se tendrá que obtener el pixel que corresponde con el par de coordenadas que caracterizan una posición y a continuación obtener el color de dicho pixel.

Se hace necesario volver a dividir este apartado, ya que para la llevarlo a cabo se precisa implementar otros procesos bastante complejos, que por separado se entienden mejor.

2.1. Obtención de un pixel con *Projection*

Como bien se ha especificado lo primero es obtener el pixel. Para ello lo que hay que hacer es utilizar, nuevamente, el API de Google que nos facilita una función por la cual se puede obtener una proyección del mapa geográfico que corresponde con la imagen que un usuario visualiza en su móvil, es decir, que mapea los pares de coordenadas y los relaciona directamente con los pixeles correspondientes de la imagen. Así si, por ejemplo, tuviéramos una imagen de 640x640 pixeles disponible en la pantalla del móvil y quisiéramos saber qué pixel ocupa en un principio la localización del usuario sobre esa imagen, que es el centro de la pantalla, a través de esa proyección podríamos obtener que el punto exacto que ocupa ese par de coordenadas es el 320x320.

Para obtener esa proyección solo se tiene que invocar al método getProjection(), de la clase MapView. Ahora bien, así como se hacía con anterioridad, para conseguir los valores de *latSpan* y *longSpan*, se volverá a esperar a tener todos los valores de la vista de nuestro mapa estabilizados, para obtener esta proyección, invocándolo en el método onResume(). Es más, debido al carácter estacionario que se desea que estos valores tengan se almacenara en la variable *projection*, ya que estos han de mantenerse fijos

aunque el jugador altere su posición hasta que la ronda cambie, momento en el que se recalcularán diferentes parámetros, entre ellos éste.

Una vez obtenida la proyección, sólo se tiene que invocar al método projection.toPixels(gPoint, out), donde *gPoint* será el par de coordenadas de la que queremos conocer la proyección, un objeto de la clase GeoPoint, y *out* es el objeto de la clase Point en la que se almacenará las coordenadas del pixel correspondientes a la pantalla del terminal del usuario. De esta manera ya se tiene el pixel.

Ahora, para obtener el color de dicho pixel se tiene que obtener de una imagen, el problema es que se tiene que obtener de una imagen estática y no de carácter dinámica como es la que el usuario obtiene del mapView, para lo que se tiene dos opciones. O bien se realiza una captura de lo que el usuario visualiza o bien se utiliza el API de Google para obtener una imagen estática de un área del mapamundi de Google Maps.

2.2.Obtención de un Mapa Estático

Se optó por la segunda opción, por la facilidad que esto suponía, ya que Google Static Maps API permite insertar una imagen de Google Maps en cualquier sitio sin ningún sistema de carga de páginas dinámicas.

Se creará el mapa a partir de los parámetros indicados en una URL que se envía a través de una solicitud HTTP estándar y que generará una imagen del mapa que podremos utilizar para nuestro objetivo. La URL de Google Static Maps API debe tener el siguiente formato:

http://maps.google.com/maps/api/staticmap?parameters

Algunos parámetros son obligatorios y otros opcionales. Como en las URL estándar, todos los parámetros se separan con el carácter ‘&’. A continuación, se muestra una lista de los parámetros y sus posibles valores.

Parámetros de ubicación:

- center (obligatorio si no hay presentes marcadores) define el centro del mapa, equidistante de todos los bordes. Este parámetro toma una ubicación con un par de coordenadas separadas por comas {latitud,longitud} (por ejemplo, "40.714728,-73.998672") o con una dirección de tipo cadena (por ejemplo "ayuntamiento, nueva york, ny") identificando una ubicación única en la superficie de la tierra.

- zoom (obligatorio si no hay presentes marcadores) define el nivel de zoom del mapa, que determina su nivel de ampliación. Este parámetro toma un valor numérico correspondiente al nivel de zoom de la región deseada.

Parámetros de mapas:

- size (obligatorio) define las dimensiones rectangulares de la imagen del mapa. Este parámetro adopta la forma de una cadena valor x valor, donde los píxeles horizontales se indican en primer lugar y los verticales en segundo lugar. Por ejemplo, 500x400 define un mapa de 500 píxeles de ancho por 400 de alto. Hay que tener cuidado porque si creas un mapa estático con dimensiones muy grandes, se reducirá automáticamente el tamaño de la imagen.

- format (opcional) define el formato de la imagen resultante. De forma predeterminada, el API de Google Static Maps crea imágenes PNG. Hay varios formatos posibles disponibles, incluidos los tipos GIF, JPEG y PNG. El formato que debes utilizar dependerá de cómo desees presentar la imagen. JPEG suele proporcionar un mayor grado de compresión, mientras que GIF y PNG proporcionan mayor nivel de detalle. En nuestro caso no se añadirá este parámetro ya que la calidad que nos da el formato por defecto es suficiente.

- maptype (opcional) define el tipo de mapa que se va a generar. Hay varios valores de tipo de mapa posibles disponibles, incluidos roadmap, satellite, hybrid y terrain. Si no se especifica el valor maptype, Google Static Maps API ofrecerá teselas de tipo roadmap de forma predeterminada.

- mobile (opcional) especifica si el mapa se mostrará en un dispositivo móvil. Los valores válidos son true o false. Los mapas que se muestran en dispositivos móviles pueden utilizar diferentes conjuntos de mosaicos optimizados para estos dispositivos. Hay que tener en cuenta que algunos dispositivos móviles como por ejemplo, los dispositivos iPhone y Android, disponen de navegadores completos que pueden mostrar conjuntos de mosaicos predeterminados sin pérdida de claridad o de precisión. En estos casos, no tienes que utilizar mobile=true. Este es el caso de nuestro proyecto, el cual está destinado a dispositivos Android, por lo que se prescindirá de este parámetro.

- language (opcional) define el idioma que se debe utilizar para mostrar las etiquetas de los mosaicos de mapas. Se trata de un parámetro indiferente para el objetivo que se busca.

Parámetros de función:

- markers (opcional) define uno o varios marcadores para adjuntarlos a la imagen en ubicaciones especificadas. Se tiene que tener en cuenta que si

se suministran marcadores para un mapa, no se tendrán que especificar los parámetros center ni zoom (que normalmente son obligatorios).

- path (opcional) define una única ruta de dos o más puntos conectados para superponerla en la imagen en ubicaciones especificadas. Al igual que el anterior parámetro se tiene que tener en cuenta que si se suministra una ruta para un mapa, no se tendrán que especificar los parámetros center ni zoom (que normalmente son necesarios).

- visible (opcional) especifica una o varias ubicaciones que deben estar visibles en el mapa, aunque no se muestre ningún marcador o ningún indicador. Se utiliza este parámetro para garantizar que se muestren determinadas funciones o ubicaciones de mapas en el mapa estático.

Notificación de parámetros:

- sensor (obligatorio) especifica si la aplicación que solicita el mapa estático va a utilizar un sensor para determinar la ubicación del usuario. Este parámetro es obligatorio para todas las solicitudes de mapas estáticos.

Para determinar que parámetros utilizar en nuestra URL, se debe tener en cuenta la función para la que se utilizará. Esta función, si recordamos, consiste en recoger el color de un determinado pixel para concretar si se trata de un color de una zona en la que podamos colocar un robot. Es por esto que se quiere que este mapa esté lo menos saturado posible, por lo que no se desean tener ni marcadores, ni rutas, ni ubicaciones. El hecho de querer prescindir de los parámetros de función, obliga a que se pongan en la ruta a diseñar los parámetros center y zoom. El resto de parámetros opcionales no serán necesarios, ya que los valores por defecto que Google Static Maps API pone son válidos para nuestra aplicación. Así que los únicos parámetros que se deben añadir a nuestra URL son los de center, zoom, size y sensor.

Aquí podemos ver una URL ya configurada para nuestro propósito:

```
"http://maps.googleapis.com/maps/api/staticmap?center=latitude,
        longitude&zoom=17&size=640x640&sensor=true";
```

Pero hay que tener en cuenta varias cosas. Por ejemplo, para centrar la imagen usaremos la posición inicial de nuestro usuario. El zoom tendrá que tener el mismo valor que el de la vista que se le ofrece al usuario del mapa dinámico. El tamaño tendrá que ir acorde con el del display del usuario.

El único de ellos que podemos asegurar desde un principio es el de sensor, puesto que se usa el GPS para determinar la ubicación del usuario, por lo tanto tendremos que ponerlo a true. Aunque, en realidad, podemos fijar también el parámetro

zoom. Por tanto los únicos valores que serán variables son size y center. A continuación explicaremos como conseguir el valor de estos parámetros.

Center

El primero de los parámetros, center, es el más sencillo de determinar. Como ya vimos con anterioridad, para llevar el mapa dinámico al centro de la posición del usuario, obteníamos la ubicación del jugador haciendo uso del método getMyLocation() sobre el objeto overlay de la clase MyLocationOverlay. Así que haciendo nuevamente uso de este objeto se puede obtener la latitud y la longitud que este parámetro necesita utilizando los métodos getLatitudeE6() y getLongitudeE6() respectivamente. El problema es que estos métodos devuelven la latitud y la longitud en un formato distinto al que tenemos que introducir en la URL. Por lo que será necesario dividirlos entre 10^6 para que cumplan el formato de la URL.

Size

El segundo de los parámetros es el que encierra una dificultad mayor, porque si bien se puede detectar con facilidad el tamaño del display que utiliza el usuario, de diferentes formas, si este tamaño supera las dimensiones establecidas por el API Static Maps de Google, el propio Google reducirá la resolución de la imagen a placer. Esto nos creará una limitación a la hora de desarrollar el juego.

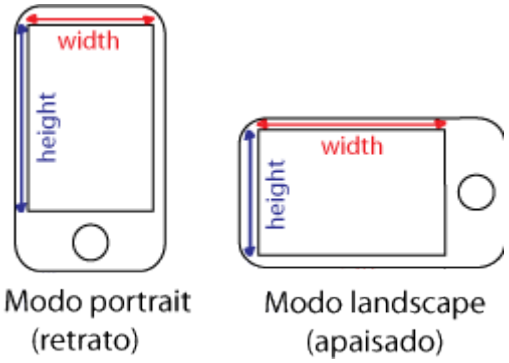
Este hecho supondrá una limitación en el área de movimiento de los androides y en el que podrán situarse los objetos de ayuda. Por esto, para solventar esta problemática se procederá teniendo unas variables en las que se almacenará el valor máximo que el API Static Maps de Google permite, 640x640. DE esta manera si las dimensiones obtenidas del display resultan mayores que estas, se sustituirán por las dimensiones límite impuestas por Google.

Si se mantiene las dimensiones impuestas por Google se tendrá que tener en cuenta a la hora de colocar los androides y los objetos teniendo en cuenta que el área será inferior al que se muestre en la pantalla y que estará centrado en la posición inicial del usuario. Esta situación se dará por hecho de ahora en adelante.

Para obtener las dimensiones de la pantalla de un dispositivo Android, es tan sencillo como, dentro de la actividad, hacer las siguientes llamadas:

```
Display display = getWindowManager().getDefaultDisplay();
int width = display.getWidth();
int height = display.getHeight();
```

Es importante destacar que este width y height son dependientes de la orientación de la pantalla, de acuerdo al siguiente gráfico:



Aunque nosotros no tendremos este problema, ya que por imposición de nuestro videojuego obligamos que la vista de modo apaisado no se pueda utilizar. Por lo que los métodos anteriormente escritos nos devolverán los valores del modo portrait.

Con todos estos parámetros ya determinados podemos configurar nuestra URL de la siguiente manera:

```
"http://maps.google.com/maps/api/staticmap?center=" +
overlay.getMyLocation().getLatitudeE6() / 1E6 + "," +
overlay.getMyLocation().getLongitudeE6() / 1E6 + "&zoom=17&size=" + width + "x" +
height + "&sensor=true";
```

De nuevo se hace necesario que este mapa se obtenga cuando los parámetros del mapView son fijos, por lo que se vuelve a realizar la obtención del mapa estático, o al menos se genera la dirección que la almacena, en el método onResume().

Http Request

Para realizar la petición http lo que se hace es, a través de una url, que se generará en este caso con la dirección que hemos configurado, se creará una HttpURLConnection, con la que se conectará con el servidor para descargar la imagen que se almacenará en objeto de la clase Bitmap y que se usará para obtener el color de un determinado pixel. Además se puede comprobar que este procedimiento se ha realizado con éxito mirando el valor que devuelve la conexión usando el método getResponseCode(), que debería ser 200 si está todo bien. Todo este procedimiento se muestra a continuación.

```
void downloadFile(String imageHttpAddress) {
        URL imageUrl = null;
        try {
            imageUrl = new URL(imageHttpAddress);
            HttpURLConnection conn = (HttpURLConnection)
                                            imageUrl.openConnection();
            conn.connect();
            int codigo = conn.getResponseCode();
```

```
            if (codigo == 200){
                loadedImage = BitmapFactory.decodeStream(conn.
                                                getInputStream());
            }
        } catch (IOException e) {
            Toast.makeText(getApplicationContext(), "Error cargando la
imagen: "+e.getMessage(), Toast.LENGTH_LONG).show();
            e.printStackTrace();
        }
    }
```

Una vez que se tiene el objeto de la clase `Bitmap` que representa nuestro mapa estático, se puede invocar al método `getPixel(x, y)` de esta clase, que nos devolverá el color del pixel de coordenadas *x* e *y*, con el que se podrá comparar con una serie de colores "prohibidos" y determinar si estas coordenadas son válidas para posicionar un androide.

2.3. Formato de Color

Esta serie de colores "prohibidos" serán aquellos colores que en un mapa de google ocupan edificios, ríos, autopistas, estadios de futbol,… En definitiva una serie de colores que corresponden con zonas de un mapa donde un transeúnte no puede acceder.

Ahora bien, estos colores no pueden seguir un formato cualquiera, es por esto que se tiene que evaluar las diferentes posibilidades que se pueden tener. Para ello acudimos a la clase *Color* de API de Android.

La clase *Color* define los métodos para crear y convertir colores a partir de enteros. Los colores se representan con paquetes de enteros, compuestos de 4 bytes: alfa, rojo, verde, azul. Además hay que tener en cuenta que los valores de la transparencia se almacenan exclusivamente en el componente alfa, y no en los componentes de color.

Los componentes se almacenan de la siguiente manera, para la componente alfa se dejan del bit 31 al 24, para el rojo del 23 al 16, para el verde del 15 al 8 y, finalmente, para el azul del 7 al 0, teniendo en cuenta que el bit cero es el situado más a la derecha. Puesto que tenemos 8bits para representar cada componente, el valor de este componente oscila entre 0 y 255, donde un 0 significa que no hay contribución de ese componente, y 255 significa el 100% de contribución.

En formato hexadecimal el opaco negro quedaría 0xFF000000, en el que la componente alfa (FF) contribuye al 100%, haciéndolo opaco, y el resto de componentes no contribuyen en ninguna medida. El blanco opaco, por poner otro ejemplo, sería 0xFFFFFFFF.

Ahora que conocemos el formato que tienen que seguir los colores que se tiene que restringir, lo único que nos queda es obtener estos colores en formato RGB. Para ello se ha hecho uso de una página de internet, http://whatsitscolor.com/,en la que se le

puede meter un color y te dice, entre otras cosas, el valor RGB de este color. Así que, realizando capturas de mapas de Google y separando los colores de zonas intransitables, se dispuso a analizar estos colores. De esta manera se pudo reconocer los valores de todos los colores pertenecientes a zonas donde un usuario no podría acceder.

A continuación, podemos ver alguno de los valores de estos colores

- 99b3cc
- 9ea09f
- d1d0cc
- eae6db
- e2e1df
- e0c27c
- c8d7d2
- e0dfdb
- d4d4d2
- c8cecc
- c6cfcc
- f9ba39
- d1960e
- df9c0d
- ffe169
- a1a1a1
- a0a0a0
- 9f9f9f
- c5d4cf
- ffc446
- afb6af
- aeb5ad
- abb6b0
- a8b7be
- 95a0a4
- 879db5
- 9cb6cf
- 95a6b0
- 8ca1bc

A todos estos valores habrá que añadirles el valor de alfa FF para que se les considere totalmente opacos.

Por último, para obtener los valores numéricos que corresponden a estos colores se tendría que invocar al método argb(int alpha, int red, int green, int blue) de la clase `Color`.

En el código de nuestro videojuego lo hemos utilizado para almacenarlo en un array que contendrá todos estos valores de la siguiente manera:

```
public int[] colores = {
         Color.argb(0xff, 0x99, 0xb3, 0xcc), Color.argb(0xff, 0x9e, 0xa0, 0x9f),
         Color.argb(0xff, 0xd1, 0xd0, 0xcc), Color.argb(0xff, 0xea, 0xe6, 0xdb),
         Color.argb(0xff, 0xe2, 0xe1, 0xdf), Color.argb(0xff, 0xe0, 0xc2, 0x7c),
         Color.argb(0xff, 0xc8, 0xd7, 0xd2), Color.argb(0xff, 0xe0, 0xdf, 0xdb),
         Color.argb(0xff, 0xd4, 0xd4, 0xd2), Color.argb(0xff, 0xcc, 0xce, 0xcd),
         Color.argb(0xff, 0xc8, 0xce, 0xcc), Color.argb(0xff, 0xc6, 0xcf, 0xcc),
         Color.argb(0xff, 0xf9, 0xba, 0x39), Color.argb(0xff, 0xd1, 0x96, 0x0e),
         Color.argb(0xff, 0xdf, 0x9c, 0x0d), Color.argb(0xff, 0xff, 0xe1, 0x69),
         Color.argb(0xff, 0xa1, 0xa1, 0xa1), Color.argb(0xff, 0xa0, 0xa0, 0xa0),
         Color.argb(0xff, 0x9f, 0x9f, 0x9f), Color.argb(0xff, 0xc5, 0xd4, 0xcf),
         Color.argb(0xff, 0xff, 0xc4, 0x46), Color.argb(0xff, 0xaf, 0xb6, 0xaf),
         Color.argb(0xff, 0xae, 0xb5, 0xad), Color.argb(0xff, 0xab, 0xb6, 0xb0),
         Color.argb(0xff, 0xa8, 0xb7, 0xbe), Color.argb(0xff, 0x95, 0xa0, 0xa),
         Color.argb(0xff, 0x87, 0x9d, 0xb5), Color.argb(0xff, 0x9c, 0xb6, 0xcf),
         Color.argb(0xff, 0x95, 0xa6, 0xb0), Color.argb(0xff, 0x8c, 0xa1, 0xbc),
         Color.argb(0xff, 0xff, 0xc3, 0x46), Color.argb(0xff, 0xf3, 0xdf, 0xac),
         Color.argb(0xff, 0xdc, 0xdb, 0xd7), Color.argb(0xff, 0xd9, 0xd8, 0xd4),
         Color.argb(0xff, 0xcd, 0xcd, 0xcd), Color.argb(0xff, 0xb9, 0xb9, 0xb9),
         Color.argb(0xff, 0xdd, 0xdb, 0xce), Color.argb(0xff, 0xc7, 0xc8, 0xc2),
         Color.argb(0xff, 0xe8, 0xe0, 0xdb)};
```

Una vez definidos los colores que tomaremos como inaccesibles, lo que se tiene que hacer es comparar el color del pixel que ocuparía la posición del androide con los almacenamos en este array, para comprobar si se trata de una posición valida.

Todo este proceso de comparar el color de nuestro pixel con los colores “prohibidos” se realiza en el método `esTransitable` que vemos a continuación.

```
private boolean esTransitable(int color) {
        boolean transitable = true;
        for(int i = 0;i<colores.length;i++){
                if(color == colores[i]){
                        transitable = false;
                        i = colores.length;
                }
        }
        return transitable;
}
```

Donde el vector colores corresponde con el vector de colores que almacena aquellos que son áreas del mapa no accesibles para un usuario, como el interior de un edificio, en el que perdería la señal GPS, o un río, imposible de acceder.

3. Visibilidad

Por último para terminar de definir todos los parámetros de los androides se tiene que definir la variable *_visible* de los mismos. Para ello se implementa otro método, esVisible(), que determina si un robot se encuentra en una perspectiva desde la que un jugador pudiera divisarlo, es decir, no se encuentra ningún obstáculo entre el androide y el usuario.

Ha este método se le tiene que pasar el punto de la pantalla del robot que se quiere determinar si es visible. Con el cual se trazará una recta hasta el punto que ocupa en ese momento el usuario, siguiendo la fórmula geométrica de la recta entre dos puntos.

$$y - y_1 = m(x - x_1)$$

Donde *m* es la pendiente de la recta y se puede caracterizar por dos puntos conocidos, en este caso la posición del androide y la del usuario del juego, como:

$$m = \frac{y_B - y_A}{x_B - x_A}$$

Se evaluarán todos los puntos que se encuentren en esa recta, en busca de alguno que coincida con los colores del vector `colores`, que si recordamos era un vector con los colores de zonas no accesibles por un jugador. De tal manera que si resulta que alguno de los colores de los puntos de esta recta coinciden con los del vector, se puede asegurar que el androide no es visible para el jugador.

Este método se ejemplifica a continuación.

```
private boolean esVisible(Point posicion) {
        boolean visibilidad = false;
        int x1;
        int x2;
        int y1;
        int y2;

        gPoint = overlay.getMyLocation();
        Point miPosicion =
                projection.toPixels(gPoint, posicion);

        if(posicion.x > miPosicion.x){
                x1 = posicion.x;
                y1 = posicion.y;
                x2 = miPosicion.x;
                y2 = miPosicion.y;
        }
        else{
                x1 = miPosicion.x;
                y1 = miPosicion.y;
                x2 = posicion.x;
                y2 = posicion.y;
        }

        for(int x=x1+1; x<x2;x++){
                int y = (x-x1)*(y2-y1)/(x2-x1)+y1;
                int color = loadedImage.getPixel(x, y);
                if(esTransitable(color))
                        visibilidad = true;
                else{
```

```
                        visibilidad = false;
                        x=x2;
                }
        }
        return visibilidad;
}
```

De esta manera se termina de definir todos los parámetros importantes de la caracterización de un robot, que sirven para caracterizar su posicionamiento.

Ítems

El procedimiento para generar los ítems es muy similar al utilizado para generar los androides, de echo la única de las diferencias que podemos encontrar en la clase es que los ítems de ayuda no tienen ninguna variable que se corresponda directamente con el pixel que ocupan en la pantalla. Esto es porque, a diferencia de los androides, los objetos de ayuda se colocan en una posición y no la varían, por lo que no es necesario tener un registro del pixel que ocupa.

1. Posicionamiento Ítems

De hecho, el resto de pasos a seguir son prácticamente los mismos.

- Primero se generan una latitud y un longitud aleatorias, teniendo en cuenta que el área en el que un ítem aparece ocasionalmente es mucho más reducida que el área de la pantalla, para que al usuario le dé tiempo de recogerlo antes de que espire los 30 segundos disponible para beneficiarse de sus efectos. En este caso el área será de un décimo las variables de *latSpan* y *longSpan*. De tal manera que la función vista anteriormente para crear las coordenadas aleatorias centradas en la posición actual del jugador quedaría, por ejemplo, para la latitud, así:

  ```
  randomLat = Math.random()*latSpan/5
  +(overlay.getMyLocation().getLongitudeE6()-latSpan /10));
  ```

- Después se evalúa el color del pixel que corresponde con el par de coordenadas generadas. Con el mismo vector de colores, de tal forma que si no coincide con ninguno, se toma como valido el valor creado, se sale del bucle y se inicializa el objeto ítem con esos valores.

- Por último se evalúa la visibilidad, aunque se espera que, debido a la proximidad del ítem, éste sea visible siempre que el jugador decida coger

la ayuda y no alejarse de ella.

Quizás la diferencia más representativa esté en el momento en que se genera. Ya que los androides se generan al inicio de una ronda, mientras que los ítems se generan uno cada cuatro minutos de juego.

Para generar el ítem cada cuatro minutos se aprovecha un hilo creado para realizar una serie de acciones cada cinco segundos, tiempo que los robots tardan en cambiar de posición, haciendo uso de un contador que se va actualizando cada vez que se ejecuta este hilo, es decir, cada cinco segundos. Cuando este contador cuenta 48 ciclos se resetea y se llama al método `generaItem()`, que crea una ayuda como se ha explicado con anterioridad. Cuando vuelven a pasar, esta vez, seis ciclos, 30 segundos, este ítem deja de tener validez, ya no se puede recoger y beneficiarse del efecto que éste activaba, el contador vuelve a cero y se espera de nuevo que llegue a 48 para generar otro ítem.

2. Funcionalidades Ítems

Como podemos ver, aparte de estos detalles a la hora de generar la posición de los objetos de ayuda, no existe gran diferencia con el posicionamiento de los androides. Pero sí que podemos observar una última gran diferencia a la hora de comparar los objetos a nivel de clase. La clase *Item* tiene una variable ID, que se trata de un valor aleatorio entre 1 y 3. Esto es porque hay tres ayudas disponibles para el jugador que como ya vimos son:

- Parar el tiempo. Permite detener la cuenta atrás, de tal forma que el jugador tenga unos segundos más para cumplir su misión. La duración de este efecto será de unos 45 segundos.

- Congelar androides. Este ítem permite paralizar temporalmente a todos los androides de una misma ronda, de esta manera el jugador goza de una ventaja adicional para completar la ronda durante un tiempo de unos 45 segundos.

- Puntos x2. Si durante el tiempo que dura el efecto de este ítem el usuario realiza alguna acción que conlleve la suma de puntos al contador, este incrementará el valor de la acción sumándose el doble del valor original de ésta. El efecto temporal de este ítem asciende al minuto y medio.

Cada una de estas funcionalidades conlleva un beneficio al usuario que se aplica durante el juego. A continuación vemos como se implementan estas funcionalidades.

Para implementar la primera funcionalidad es necesario hacer una pequeña introducción de cómo se contabiliza la cuenta atrás, que se explica más detenidamente llegando al final del punto del punto 4.3.3. Se hace mediante un hilo en el que se va incrementando un contador cada 1000 milisegundos, es decir cada segundo.

Por tanto para detener el contador durante 45 segundos no es necesario paralizar el hilo que incrementa este contador, sino que basta con deshacer el incremento que se realiza en este hilo. Para ello se aprovecha el hilo que mueve los androides que, como veremos se ejecuta cada 5000 milisegundos, y restamos al contador de tiempo 5 ciclos, hasta que se cumpla, los 45 segundos, es decir que esta acción se realizará 9 veces.

Para implementar la segunda funcionalidad habría que detener el hilo que controlase la rutina de los movimientos de los androides, sin embargo optamos por saltarnos la sentencia que mueve a los robots si se ha cogido este ítem, de esta manera podremos paralizar los androides.

Pero nuevamente hay que controlar 45 segundos, durante los que durará el efecto de esta ayuda. Como se vio en el primer ítem la sentencia de mover los androides ocurre cada 5 segundos, por lo que nuevamente con saltárnoslo 9 veces contabilizaremos los 45 segundos necesarios.

Por último la última ayuda suma a la puntuación total el doble de los puntos conseguidos por algunos sucesos, como, por ejemplo, capturar un androide. Para ello basta con activar una variable booleana que implica que cada vez que ocurre uno de estos sucesos se sume el doble de puntos.

Una vez más, para controlar el minuto y medio en que esta ayuda estará activada, se aprovecha el hilo que controla la rutina de mover los robots, de tal manera que con la captura del ítem se activará una variable, que nos indicará que estos valores se tienen que ver multiplicados por 2 al ocurrir ciertos sucesos, y se desactivará cuando se completen este tiempo, 18 ciclos del contador.

Una vez explicadas como se implementan las funcionalidades de los diferentes objetos de ayuda, se termina de explicar las diferencias entre los ítems y los androides, a la hora de crearlos.

Marcadores

Pero tenemos que volver ahora a un punto que ambos tiene en común. Los marcadores que señalan su posición sobre la vista del mapa disponible para el usuario.

Para generar los marcadores se ha creado una clase llamada MyOverlay. Puesto que la funcionalidad para ambos es exactamente la misma, señalar en el mapa la posición que se ha generado con anterioridad para que el usuario disponga de una guía,

no es necesario hacer una distinción entre los marcadores y por tanto una clase es suficiente.

Pero sí que existe una diferencia en la imagen que se pondrá en el mapa, por lo que sí que se tendrá que hacer una pequeña distinción en la clase. Para hacer esta distinción de la imagen se utiliza, en el constructor, un identificador, de esta manera se puede inicializar la imagen que se quiera cargar en el mapa en función de lo que se quiera mostrar en el mapa. Es decir que a la hora de crear un marcador se le tendrá que pasar un 0 si se quiere que el marcador sea un robot o un 1 si se quiere que sea un ítem. Además del identificador en el constructor habrá que indicarle el punto geográfico en el que estará situado este marcador.

El resto de la clase resulta bastante sencilla. Lo único que se tiene que hacer es sobrescribir el método draw() de la clase Overlay, de la cual extiende la clase MyOverlay. De esta manera se puede tener, además de la funcionalidad que éste método facilita, la posibilidad de obtener el pixel sobre el que se pintará el marcador y centrar la imagen en él, como podemos ver a continuación.

```
@Override
public boolean draw(Canvas canvas, MapView mapView, boolean shadow,
long when) {

        super.draw(canvas, mapView, shadow);

        Point scrnPoint = new Point();
        mapView.getProjection().toPixels(_point, scrnPoint);

        Bitmap marker =
               BitmapFactory.decodeResource(mapView.getResources(),
drawable);

        canvas.drawBitmap(marker, scrnPoint.x - marker.getWidth() / 2,
scrnPoint.y - marker.getHeight() / 2, null);

        return true;
    }
```

Este método se llamará cada vez que se añada un marcador a la lista de `Overlays` que el mapa de Google tiene. Para añadirlo lo único que tenemos que hacer es invocar la siguiente sentencia.

```
mapOverlays.add(markerItem);
```

Donde *mapOverlays* proviene de la sentencia `mapOverlays = map.getOverlays()`.

Estos marcadores sufren alteraciones en el mapa. Por ejemplo, el marcador de los objetos de ayuda desaparece si se le coge o expira el tiempo para recogerlo y

beneficiarse de sus efectos. En el caso del marcador de los androides ocurre lo mismo si se les captura desaparecen y continuamente están en movimiento, lo que supone estar alterando el pixel en el que se encuentra continuamente.

Para atender esta funcionalidad de eliminar el marcador que previamente se había situado en un pixel determinado del mapa, se vuelve a hacer uso de la lista de `Overlays` disponibles en instancia de `mapView` que se ha llamado *map*. Simplemente con ejecutar una sentencia, el sistema busca el marcador a borrar y lo elimina. Esta sentencia es la siguiente.

```
mapOverlays.remove(markerItem);
```

Donde `markerItem` es el marcador que se desea eliminar, que en este caso corresponde con el de un ítem de ayuda.

Si bien para eliminar un marcador esta sentencia es válida, existe una situación en la que habría que utilizar otra. En la que los robots se mueven, ya que estos marcadores no se eliminan si no que son sustituidos por otros con una posición diferente. En este caso vale con tener localizada la posición en la que se almacenó el marcador de un androide y sustituirlo con otro con la sentencia mapOverlays.set(posicion, marker), donde *posicion* es la posición que ocupará el nuevo marcador.

En ambas acciones habrá que refrescar la imagen que al usuario se le muestra por pantalla. Este se hace con el método .invalidate(), que se llama sobre el mapa, objeto de la clase `mapView`, que se dispone.

Para terminar con la funcionalidad de los elementos virtuales externos con los que el usuario puede interactuar, androides e ítems, solo queda explicar una funcionalidad de los robots. La inteligencia artificial que controla su movimiento.

Movimiento de los Androides

Para controlar el movimiento de los robots se les ha dotado de una inteligencia artificial que, a diferencia de la que se puede encontrar en otros videojuegos, es extremadamente sencilla. Esta sencillez se debe, entre otras cosas, a que no es objeto de este proyecto, ni siquiera del otro encargado de completar las funcionalidades del juego, el desarrollar una inteligencia artificial que controle la rutina del movimiento de los robots. Así como en juegos conocidos como el *Pacman* podemos encontrar que cada uno de los fantasmas tiene una rutina diferente que controla sus movimientos, en el que se desarrolla para este proyecto es mucho más simple.

Por decirlo de alguna manera, los androides tienen un área en el que moverse cada cinco segundos y pueden decidir la dirección en la que se mueven. Ahora bien sólo

se moverán si el punto al que deciden moverse es transitable, sino lo es, su desplazamiento será nulo y se quedarán en el punto en el que estaban.

Por tanto se desprenden varias cosas que se tienen que tener en cuenta a la hora de controlar el desplazamiento de los androides:

- Habrá que controlar el tiempo que transcurre entre movimientos.
- Se tendrá que establecer un área de movimiento por el que un robot pueda desplazarse.
- Habrá que controlar la posibilidad de que el androide falle en su elección y por tanto se quede quieto.

Para controlar el movimiento periódico de los androides, que se ejecuta cada cinco segundos, se creará una rutina que se esté ejecutando en paralelo a otros procesos del videojuego. De esta manera no se desatenderá ninguna de las funcionalidades del videojuego. Para ello se utilizará un hilo y un manejador que lo controle.

Este hilo se ejecutará cada 5000 milisegundos, cumpliendo así con las expectativas de que el robot tiene que tener la posibilidad de moverse cada 5 segundos.

A continuación se muestra el código del hilo que se genera, en cuyo interior se ejecuta el método que moverá a todos los androides.

```
final Handler mHandler = new Handler();

protected void mueveDroids(){
        Thread t = new Thread(){
                public void run() {
                        while(!Ronda.hamsters.isEmpty()){
                                try{
                                        Thread.sleep(5000);}
                                catch(InterruptedException e){
                                        e.printStackTrace();
                                }
                                mHandler.post(mueveDroid);
                        }
                }
        };
        t.start();
}
```

Como se puede observar con dormir el hilo 5000 milisegundos basta y después de esta ejecución se llama al método que en realidad mueve todos los androides. Con esto salvamos el primer problema que se platea a la hora de mover los robots, de hacer que el movimiento de los robots sea periódico.

Ahora el problema que se nos plantea es el proporcionar un área de maniobra para los enemigos, el cual se proporcionará dentro de mueveDroid, que es un objeto de la clase Runnable. Además dentro de mueveDroid se implementará la última condición que se nos plantea para hacer que los enemigos sean autómatas, controlar la posibilidad de que el androide falle.

Para cumplir la primera de las condiciones que nos queda cogeremos la posición geográfica que actualmente está ocupando el androide y se le sumará a cada una de las coordenadas un valor que puede variar entre - factorMovimiento y factorMovimiento, que es una variable que indica el desplazamiento máximo que un robot puede tener. De esta manera se genera un cuadrado de (2x factorMovimiento)^2 de superficie centrado en la posición del robot, por el que este se puede desplazar.

Ahora se generará, de manera aleatoria el valor que el androide se desplazará en cada una de las coordenadas y se les sumará a la coordenada que le corresponda. Pero queremos excluir el caso en el que se quede quieto, ya que este caso corresponderá al caso en el que la posición que elija impida su movimiento por tratarse de una zona inaccesible. Además se quiere imponer que el máximo desplazamiento que pueda llevar acabo el androide sea también el mínimo, es decir, que no exista el desplazamiento cero, sino, únicamente, - factorMovimiento y factorMovimiento.

Con estas nuevas condiciones, como se tiene que proceder es generando una número aleatorio entre el -1 y el 1, que redondearemos y si coincide con que es 0, se repite la operación. Una vez tengamos este valor, es cuando se multiplica por el valor impuesto por factorMovimiento, y será este valor el que se sume a las coordenadas del androide. Todo este proceso se puede ver a continuación.

```
int latitud =(int)droid.getLatitud();
int longitud = (int) droid.getLongitud();

int desplazamientoX = 0;
int desplazamientoY = 0;

while(desplazamientoX == 0)
        desplazamientoX=(int)Math.round((Math.random()*2 - 1));
while(desplazamientoY == 0)
        desplazamientoY=(int)Math.round((Math.random()*2 - 1));

latitud = latitud + desplazamientoX* droid.factorMovimiento;
longitud = longitud + desplazamientoY* droid.factorMovimiento;
```

Con estas condiciones que se han impuesto se puede observar que el desplazamiento del robot se limita a cuatro puntos:

- (latitud + factorMovimiento ; longitud - factorMovimiento)
- (latitud – factorMovimiento ; longitud - factorMovimiento)

- (latitud + factorMovimiento ; longitud - factorMovimiento)
- (latitud – factorMovimiento ; longitud - factorMovimiento)

de los cuales tiene que elegir uno.

Por último se tiene que evaluar, como ya se había hecho con anterioridad, si la posición que pasará a ocupar será una posición accesible, que en el caso de no serlo se habrá perdido la oportunidad de moverse.

Para ello se tiene que obtener la proyección de la posición geográfica que puede ocupar el androide y, a partir del pixel que se logra, obtener el color a través del mapa estático descargado al principio. Si resulta que la posición es válida tendremos que sustituir los valores del androide y la posición que ocupaba en la lista de los marcadores, por el nuevo marcador que indicará su localización en la vista del mapa disponible para el jugador. Todo este proceso se ejemplifica a continuación.

```
gPoint = new GeoPoint(latitud,longitud);
posicion = projection.toPixels(gPoint, posicion);

x = posicion.x;
y = posicion.y;

color = loadedImage.getPixel(x, y);

if(esTransitable(color)){

        droid.setPosicion(posicion);

        droid.setLatitud(gPoint.getLatitudeE6());
        droid.setLongitud(gPoint.getLongitudeE6());

        Ronda. droids.setElementAt(droid, i);

        MyOverlay marker = new MyOverlay(gPoint,0);
        mapOverlays.set(i+1, marker);
        map.invalidate();

}
```

Con esto se implementa toda la rutina que maneja el movimiento de los androides, que como se anunció no era nada enrevesado.

Niveles

Pocos aspectos funcionales quedan por describir en lo referente a la implementación del juego. Pero hay algo que está estrechamente relacionado con un videojuego y que aún no hemos abordado. Los niveles.

Si bien hemos hablado de ellos, en el sentido de que si se capturan todos los androides de una ronda se pasa automáticamente a la siguiente ronda, no hemos comentado los aspectos más importantes que caracterizan una ronda. Hablaremos de cuantos androides hay por ronda, de cuando se cambia de ronda y cómo se le indica al jugador que ha completado una ronda y pasa a la siguiente, cuantas rondas contiene el juego y cómo se indica el final del juego.

Una ronda se caracteriza por el número de androides a capturar, por ello, cuando se capturan todos los androides se pasa a la siguiente ronda. Pero ¿cuántos androides hay en cada ronda? Para que el número de robots por ronda fuera creciente se diseñó un algoritmo que generara posiciones aleatorias para colocar los androides en orden dos veces la ronda en la que el usuario se encontrara más uno. De esta manera, si se encontraba en la primera ronda el número de androides fueran 3, en la segunda 5, en la tercera 7 y así sucesivamente. Este algoritmo solo habría que meterlo en un bucle como podemos ver en la siguiente sentencia:

```
for(int i=0; i<(Ronda.ronda*2+2);i++){
        .
        .
        .
}
```

Dentro del bucle se colocaría las líneas de código necesarias para generar una posición aleatoria y convertirla en un objeto `Droid`, que se han visto con anterioridad.

Para completar una ronda tenemos una condición indispensable, que se hayan capturado todos los androides de esta. Es por esto que, cuando se captura un androide y se indica en la clase *Ronda*, se debe comprobar siempre si éste era el último que quedaba de esta ronda, para tomar las medidas oportunas. Esta operación se hará en dos métodos con sendas llamadas a un par de métodos que realizan esta función en la clase Ronda:

```
public static void capturaDroid (int posicion){
        droids.elemetAt(posicion).capturado=true;
        …
}


public static void incrementaRonda(){
        ronda++;
        pasoRonda = true;
        …
}
```

Cada objeto de la clase Droid de cada ronda, almacenados en el vector de *droids* de esta clase, tendrá un booleano que indicará si ese androide ha sido capturado. De esta

manera podremos eliminar el marcador en la actividad principal del juego y comprobar si era el último androide.

```
            if(droid.capturado){
                            mapOverlays.remove(i+1);
                            Ronda.droids.remove(droid);
                            droid.borrar();
                            map.invalidate();
                            if(Ronda.droids.isEmpty())
                                Ronda.incrementaRonda();
            }
```

Como podemos ver el comprobar si se trataba del último robot de la ronda depende de si el vector *droids*, perteneciente a cada ronda, está vacio. En caso afirmativo, se llama a incrementa ronda que, además de incrementar el número de la ronda, pone a true la variable booleana *pasoRonda*.

Para cambiar de ronda basta con invocar el método onResume() de las Activity de Android, ya que es en este método en el que se generan las rondas o, mejor dicho, en el que se inicializan el posicionamiento de los androides con la llamada al método initDroidLocation(). Por lo que, si uno se da cuenta, el generador de rondas no existe nada más que como concepto.

En cambio sí existe una clase *Ronda* que contiene, generalizando para todas las rondas, todas las funcionalidades y parámetros oportunos de una ronda.

El último aspecto que queda por ver en lo referente a las rondas son los avisos que se le muestran al usuario cuando se genera una nueva ronda. Este aviso se mostrará en forma de mensaje en la parte de debajo de la pantalla, de esta manera no se impide la continuidad del juego, ya que el tiempo sigue corriendo. Para ello las actividades tienen un elemento que permite realizar esta funcionalidad, Toast.

Ahora bien, que tendremos que hacer diferencia entre el mensaje que se muestra desde la primera ronda y el que se muestra en la última ronda. Mostrándolo como se indica a continuación.

```
        if(ronda!=10)
                Toast.makeText(getApplicationContext(),"Ronda " + 0 +
ronda , Toast.LENGTH_LONG).show();
        else
                Toast.makeText(getApplicationContext(),"Ronda Final",
Toast.LENGTH_LONG).show();
```

Para mostrar la pantalla de fin del juego, se hará uso de la clase *AlertDialog*, la cual nos permite mostrar una ventana con información al usuario. Se tendrá que diferenciar entre la pantalla de completar con éxito el juego y la de límite de tiempo expirado. Pero en definitiva el procedimiento es el mismo.

```
AlertDialog.Builder alert = new AlertDialog.Builder(this);
alert.setMessage("Do you want to close this window ?");
alert.setTitle("Title");
alert.setIcon(R.drawable.icon);
alert.show();
```

Llegados a este punto solo quedan por explicar dos funcionalidades que se añaden a la aplicación para dar un aspecto todavía más cercano a un videojuego si cabe.

Puntuación

La primera de ellas es la puntuación que un jugador puede obtener a lo largo del videojuego, que se irá incrementando con diferentes sucesos, así como la captura de un androide, la recogida de un ítem o el paso de ronda, o con el tiempo restante tras completar el juego.

Para ello existirá un contador, denominado *puntuacionTotal*, en el que se irán sumando los puntos obtenidos por cada una de estas ocurrencias. Los puntos se incrementarán en cada uno de los métodos que sirven para indicar cada una de las ocurrencias que se han señalado antes, es decir en `capturaDroid()`, `recogeItem()`, y `incrementaRonda()`, respectivamente. De la siguiente manera.

```
puntuacionTotal = puntuacionTotal + puntuacion;
```

Lo único que difiere de una sentencia a otras es el valor de `puntuación`, que si se captura un androide sumará 750 puntos, si se recoge un ítem sumará 250 puntos y si pasa de ronda sumará 500 puntos.

En el caso de los puntos conseguidos por el tiempo que resta al final se contabilizarán una vez se detecte que el jugador ha completado el juego, comprobando que la ronda es >10, y se sumarán 600 puntos por cada 10000 milisegundos, es decir, 600 puntos por cada 10 segundos, de tal manera que si el tiempo restante es de un minuto podrá acumular hasta 6000 puntos extras. Esta puntuación es tan alta por la dificultad que conlleva completar el juego.

Una vez contabilizados los puntos se mostrará el letrero de fin del juego con la puntuación conseguida.

Temporizador

Por último se explicará el procedimiento seguido para contabilizar la cuenta atrás con la que se anuncia el fatídico final del juego. Y tras el cual se muestra la pantalla de GameOver.

Realmente no es necesario que se trate de una cuenta atrás, aunque realmente al usuario se le muestre como una cuenta atrás. Bastaría con generar una cuenta que llegara a 15 minutos, suponiendo que el usuario no lograra exceder de este límite al pasar de ronda, en cuyo caso la cuenta ascendería a más minutos.

Es tan sencillo como generar un hilo que cada 1000 milisegundos incrementara un contador que hiciera las veces de timer. De esta manera si el contador llegara a 900, quince minutos, se daría por finalizado el juego.

Pero hay que tener en cuenta, como se ha dicho ahora mismo, que al pasar de ronda el usuario puede sumar algo más de tiempo del que se da inicialmente, sobre todo en las primeras rondas, en las que el número de androides es muy inferior al que se puede encontrar en los últimos niveles.

Lo único que hay que hacer es, cuando se produzca un cambio de ronda, es restar a este contador el tiempo de bonificación que asciende a 5 minutos, en total 300 ciclos. De tal manera que si resulta un número negativo tendrá más tiempo para llegar al límite fijado en 900. Por ejemplo, el jugador que acaba de finalizar la primera ronda sólo ha consumido dos minutos y medio, por lo que el contador está a 150, al restársele los 300 ciclos el contador tendrá el valor de -150, por lo que tendrá para completar las siguientes rondas un total de 1050 ciclos que equivalen a diecisiete minutos y medio.

Estas cuentas se realizarán en el momento en que se active una variable booleana que indica que el juego ha terminado con éxito, y la cual indicará que todas las funcionalidades del juego han de detenerse y llevarse a cabo el recuento de puntos para mostrar por pantalla la puntuación conseguida.

Con este apartado se puede dar por finalizada la explicación detallada de la implementación y el desarrollo completo de este proyecto.

Capítulo 6

Conclusiones

Extraer los aspectos positivos y negativos de una aplicación móvil, como es este proyecto, pasa necesariamente por evaluar el grado de cumplimiento de los objetivos inicialmente marcados y comprobar si se han conseguido una vez finalizado el mismo.

Así pues, a continuación se enumeran de nuevo esos objetivos, adjuntando a estos las conclusiones y el grado cumplimiento de los mismos.

- Diseño de la lógica de un videojuego que permita el uso de las diferentes tecnologías presentes en los terminales Smartphone
- Uso del GPS como recursos de los terminales Smartphone
- Uso de la librería creada en un proyecto que se desarrolla paralelamente a éste
- Uso de las librerías de Google

1. **Diseño de la lógica de un videojuego que permita el uso de las diferentes tecnologías presentes en los terminales Smartphone.** Se puede decir que este requisito es el que se ha buscado cubrir con mayor esfuerzo y el que con mayor certeza se puede afirmar haber cumplido. Ya que desde un principio el diseño del videojuego se había fundamentado en el uso de las tecnologías más punteras en los terminales Smartphone. Prueba de ello es que la lógica del videojuego hace constante uso de posicionamientos, orientaciones y pulsaciones sobre la pantalla del terminal, algo que no podría haberse cumplido si no se hace uso de los sensores integrados en cualquier terminal de estas características, de lo que derivan otros requisitos del proyecto. Sin la incorporación de estas tecnologías en el juego sería imposible concebirlo, ya que los datos que facilitan son imprescindibles para el posicionamiento de los androides e ítems, visualizarlos y capturarlos, y como consecuencia de ello son imprescindibles para la consecución del juego. La integración de los datos extraídos de estos sensores en la lógica del juego ha significado un punto muy importante a la hora de dotar al videojuego de dinamismo y cierta realidad. Esto, junto a otros aspectos indiscutibles en cualquier videojuego como puede ser una interfaz intuitiva y fácil de manejar, hacen que esta aplicación no solo cumpla este requisito expuesto, sino que además sea un juego divertido y con gancho.

2. **Uso del GPS como recursos de los terminales Smartphone.** Igual que afirmamos el cumplimiento del primer requisito, tenemos que hacerlo con éste, al tratarse consecuencia directa de la lógica del juego. Puesto que se hace necesario el posicionamiento, tanto del usuario de la aplicación, como de los diferentes objetos, se convierte éste en un requisito indispensable para el correcto funcionamiento del juego. Es por esto que el uso de este sensor de captación del posicionamiento del usuario por vía satélite fundamenta las bases del proyecto, ya que, sin un correcto posicionamiento del usuario, teniendo en cuenta que el rango de error del posicionamiento fuera lo suficientemente pequeño, no se podría asegurar la continuidad del juego sin fallos del GPS y tampoco se podrían posicionar los diferentes objetivos a capturar por el jugador dentro de un rango asequible y que permitiera la jugabilidad de la aplicación.

3. **Uso de la librería creada en un proyecto que se desarrolla paralelamente a éste.** Mediante estas librerías se permite introducir gráficos OpenGL en el videojuego a partir de las localizaciones geográficas que se obtienen gracias al GPS. Con estos gráficos podemos dotar a la aplicación creada de gráficos 3D, dando así al videojuego un aspecto más realista que permite al usuario interactuar con objetos que se fusionan con el mundo de una manera más natural que en lugar de si se trataran de gráficos planos. Además el hecho se usar esta librería permite integrar al videojuego el uso de los acelerómetros. Usando los acelerómetros es como se conoce que el usuario está orientando el móvil hacia donde virtualmente está colocado el objeto y mostrar el objetivo sobre la imagen captada de la cámara del terminal del jugador. Por otro lado estas librerías permiten controlar todo tipo de eventos táctiles, como los que se producirían al capturar uno de los androides objetivos del juego. Por último, comentar que asociados a estos eventos táctiles existe un efecto sonoro que acompaña la pulsación sobre el objeto a capturar. Todo esto es lo que logramos hacer como consecuencia de utilizar estas librerías en el proyecto, aumentando con todo ello la experiencia del usuario y dotándolo de un significado más próximo a lo que entendemos por realidad aumentada. Se puede afirmar que el cumplimiento de este requisito ha sido total, aplicándolo precisamente de la manera que se esperaba. Los pares de coordenadas que se facilitan son las del usuario de nuestro juego y las aleatorias que se generan a partir de ésta para los diferentes objetivos. De esta manera se logra que a medida que nos acerquemos o alejemos de los diferentes objetivos, estos vayan creciendo o disminuyendo en la visualización sobre la cámara del terminal, siempre que estén dentro del campo de visión del usuario y no haya ningún obstáculo de tamaño considerable, como un edificio, impidiendo la visión directa.

4. **Uso de las librerías de Google.** Este requisito, que en primera instancia pudiera parecer secundario por el hecho de presentarse como herramienta de apoyo, se convierte en uno de los más importantes para la consecución del juego. Las

librerías de Google no solo nos permiten presentar la información de los objetivos sobre un mapa al jugador de la aplicación de una manera sencilla e intuitiva, mejorando la interfaz de usuario y con ello la experiencia del mismo, sino que además se convierten en una herramienta clave para determinar el correcto posicionamiento de los androides y de los ítems, ya que, gracias a las imágenes que se consiguen con los mapas estáticos, podemos determinar si colocamos los diferentes objetivos en un lugar accesible para el usuario de nuestra aplicación. Además el api de Google permite mapear todas las coordenadas que se presentan en la pantalla del terminal móvil, de tal manera que se puede determinar el pixel que ocupa en esa misma pantalla. Esta cualidad se convierte en fundamental a la hora de colocar los objetivos sobre el mapa que al usuario se le muestra. Por tanto podemos asegurar que le cumplimiento de este requisito se consigue con superando las expectativas planteadas en primera instancia.

Se hace necesario dedicar un apartado a la valoración personal, no tan centrada en los objetivos propios que surgieron, sino en la realización del proyecto y todo lo que éste conlleva.

Para empezar me gustaría señalar lo enriquecedor de la experiencia, ya no sólo por el hecho de que la realización de un PFC te permite acercarte, con bastante acierto, a los proyectos profesionales a los que te enfrentas en el día a día en la vida laboral, sino porque además, a diferencia de otros proyectos, en éste hemos podido conocer las ventajas y desventajas del trabajo en equipo.

Empezando por este último aspecto puedo afirmar que trabajar en equipo te da cierta seguridad a la hora de desarrollar un proyecto de este calibre. Ya que cuando algo que a ti se te puede atragantar tu compañero puede orientarte, con un rol muy cercano al del tutor, pero con una implicación mucho mayor, puesto que trabaja para un mismo fin y en un mismo proyecto. Pudiendo nutrirte de esta manera de la propia experiencia de tu compañero en la resolución de problemas, consultas a foros especializados e incluso de su propio trabajo. Pero, como todo, también tiene sus puntos menos favorables. El hecho de trabajar en equipo no surge de la comodidad, sino de la necesidad que genera el querer abarcar un proyecto muy grande. El trabajo en equipo te permite centrarte en un aspecto del proyecto en lo que tu compañero sé dedica a otro. Pero esto es un arma de doble filo, ya que por norma general las dos partes están estrechamente relacionadas y son bastante dependientes. Esto se convierte en el mayor inconveniente del trabajo en equipo. Puesto que, al depender una parte de la otra, un retraso en una de las partes conlleva un retraso equiparable en la otra. Coordinar el trabajo para evitar estos problemas se hace realmente difícil y el éxito de esta coordinación radica en una meticulosa programación.

Capítulo 6

Conclusiones

Aún con una meditada programación, todo proyecto es susceptible a cambios de requisitos y fechas, y más en nuestro caso, en el que gran parte del trabajo lo hemos compaginado con prácticas y el curso de adaptación al grado. Aquí nos ha favorecido el hecho de que no se tratara de un cliente real y que no estuviéramos sujetos a fechas de entregas, a pesar de que el objetivo inicial era presentar a finales de octubre del año pasado. Aunque en la fecha de la defensa también influyeron otros aspectos como el cambio de objetivos durante el transcurso del proyecto.

Esto último es quizá uno de los puntos que más confluyen con la realidad, ya bien sea por los cambios de los objetivos por la imposibilidad de satisfacer los requisitos de alguna manera, o bien por la volatilidad de la idea del cliente en el resultado final. Es muy normal que la idea inicial de un cliente cambie a lo largo del proyecto, porque la idea no esté bien definida en un principio, o simplemente porque quiera añadir funcionalidades o eliminar otras que pudieran parecer útiles cuando se pensaron y luego no lo fueran. De igual manera puede ocurrir que, los que estemos en el otro lado para satisfacer las necesidades del cliente, tengamos que buscar alternativas para llegar a un mismo punto o incluso eliminar algunos aspectos por la imposibilidad de resolverlo. Precisamente todo esto la experiencia enriquece nuestra capacidad de respuesta, la tolerancia a los cambios y rapidez de reacción, todos ellas características que nos preparan para el futuro laboral.

A parte de todo esto, este proyecto daba la oportunidad de asentar conocimientos en el ámbito de java, pero con el matiz de aplicarlos a la última tecnología móvil y desarrollando sobre un sistema operativo Android, lo que te permite al tiempo adquirir conocimientos muy diversos y amplios sobre una materia totalmente actual.

Por último comentar que con este trabajo se ha desarrollado un videojuego que en un principio pudiera parecernos imposible de realizar. Pero que a día de hoy se puede mirar con orgullo el resultado, sabiendo que hemos cumplido con los objetivos, tanto técnicos como personales, aún quedando mucho trabajo por realizar.

Capítulo 7

Trabajos Futuros

Una vez terminado éste proyecto se sabe que existe un margen de mejora en muchos sentidos, así como con amplias posibilidades de ampliación por medio de extensiones de funcionalidad o utilidades complementarias. A continuación se listan algunas de las posibles formas de darle continuidad.

- Mejorar inteligencia artificial de los androides
- Introducir más funcionalidades en los ítems
- Mejorar la interfaz gráfica del usuario
- Introducir hilo musical durante el juego
- Mejorar el posicionamiento satelital de los usuarios
- Permitir todas las funcionalidades de dinamicidad que Google da a los mapas
- Introducir al usuario más en la historia del juego
- Estandarizar el videojuego con independencia de la versión del S.O.

1. **Mejorar inteligencia artificial de los androides.** Este es un punto que al atenderlo se daría una mayor dificultad al videojuego y que le dotaría de además de un aspecto más realista. Esto se debe a que los androides que el jugador tiene que capturar se mueven de manera aleatoria, incluso en ocasiones se quedan parados, si van a moverse por ejemplo dentro de un edificio. Podría hacerse un sencillo algoritmo que les hiciera moverse en la dirección opuesta a la que el usuario de la aplicación se encuentra, permitiéndoles huir de su perseguidor y no dejando tan fácil su captura.

2. **Introducir más funcionalidades en los ítems.** Este aspecto es uno que a la hora de diseñar el juego se pensó, dando una mayor variedad de ítems de apoyo al usuario. Finalmente no se implementaron, debido a que realmente no era el objetivo final del proyecto, sino que una opción que se quiso explotar para darle a la aplicación un aspecto más cercano de lo que podemos conocer cualquiera como un juego. Entre las opciones que se barajaban implementar se

encontraban, aparte de las que se han implementado, las siguientes:

- Aumentar el rango de captura de los robots, así el usuario no tendría que encontrarse a menos de 7 metros para poder capturar los objetivos.

- Ralentizar el avance de los robots, de esta manera no se les pararía por completo, pero sí que la distancia media que recorrerían sería menor.

- Visualizarlos en la cámara a través de los edificios, lo que permite situar a un robot a pesar que un edificio se encuentre entre el usuario y el objetivo a capturar.

- Aumentar el rango de visión del usuario, el cual solo llega a 20 metros, por lo que sólo si se acerca a esta distancia visualiza el androide en la cámara.

- Brújula indicadora del androide más próximo, lo que permite al jugador orientarse sin necesidad de mirar el mapa.

- Marcar la distancia de los objetivos sobre el mapa, de esta manera se facilita la decisión del usuario de qué androide ir a capturar.

3. **Mejorar la interfaz gráfica del usuario.** Actualmente no se ha cuidado mucho la estética de la interfaz gráfica, ya que no suponía un objetivo propio del proyecto. En cambio tener una interfaz más llamativa haría que el juego fuera más atractivo de cara a jugarlo, sería un aspecto que sin duda habría que atender si se deseara comercializar la aplicación. Además de generar un GUI multilingüe que atraería aún más en el mercado internacional.

4. **Introducir hilo musical durante el juego.** Todo buen videojuego que se precie de ello tiene una banda sonora inconfundible, no sólo efectos sonoros. Además de dar un hilo musical característico, podría servir de ayuda para poner en una mayor tensión al jugador y avisarle incluso con un hilo musical más frenético a medida que la cuenta atrás llegara a 0.

5. **Mejorar el posicionamiento satelital de los usuarios.** Se trata de un punto que mejoraría significativamente la jugabilidad de la aplicación, ya que a pesar de no permitir el inicio del juego sin una precisión mínima o incluso interrumpirlo si, una vez iniciado el juego, se perdiera la precisión, la exactitud con la que el GPS posiciona al usuario no siempre es constante y en ocasiones lo ubica en otro sitio en el que no se encuentra, lo que afecta a la continuidad de la partida. Este es un factor que atañe más bien a la captación de satélites y a las reflexiones que los edificios provocan sobre la señal del GPS, una posible línea futura podría

centrarse en realizar un estudio de cómo combatir estos problemas de tal manera que una aplicación en tiempo real de posicionamiento no ocasionara estos problemas.

6. **Permitir todas las funcionalidades de dinamicidad que Google da a los mapas.** Estas funcionalidades de los mapas dinámicos es algo que para el correcto funcionamiento del juego se tuvo que bloquear, ya que daban lugar a fallos de la aplicación. El mayor problema se encontraba al hacer zoom o mover el mapa dinámico, ya que se rompía la coincidencia con el mapa estático que se utilizaba como base para posicionar y mover los androides, un problema que tendría fácil solución simplemente descargando el nuevo mapa estático con las nuevas dimensiones o con el nuevo centro del mapa. Pero ocurría también, que al dejar un androide fuera del mapa que el jugador visualiza en la pantalla de su móvil, no se podía obtener el pixel que ocupaba en la imagen estática ese objetivo, porque, aunque las coordenadas permanecieran inalteradas, no aparecían en el móvil.

7. **Introducir al usuario más en la historia del juego.** Esta línea de trabajo futuro se debe a que, aunque en la memoria del proyecto se deja de manifiesto la historia que motiva la creación del juego, en la aplicación no se hace ninguna referencia a la misma. Podría hacerse más participe al jugador añadiendo en el menú de inicio una opción que te permitiera leer la historia o incluso mediante alguna cinematográfica al inicio de una nueva partida y al final de las mismas.

8. **Estandarizar el videojuego con independencia de la versión del S.O.** Este último apartado se debe sobre todo a restrictividad de la versión del sistema operativo para el que se ha diseñado el videojuego (Android 2.2). Muchas de las funcionalidades que el videojuego adquiere mediante la programación se consiguen de manera diferente para las diferentes versiones, por lo que si se intenta ejecutar este en móviles de versiones distintas no se garantiza su funcionamiento correcto. Por ejemplo para versiones inferiores de esta se ha comprobado que la pantalla en la que se muestra la captura de la cámara se vería girada, y que otras funcionalidades fuerzan el cierre de la aplicación. Para versiones superiores a esta, únicamente se ha podido probar en algunas de la familia 2.x y no se han observado cambios significativos, aunque la muestra no es representativa por su escasez.

Bibliografía

- Sistema de Realidad Aumentada para Aplicaciones Android, PFC Natalia Mercedes Fernández Sánchez.

- Programación Android. Pedro Dans Alvarez, traductor. Ed Burnette. Anaya Multimedia.

- G. Fernandez Arnedo, J.J. Toledano Mancheno. Navegación General y Radionavegación. Editorial AVA.

- Vladimir Silva. Pro Android Games. Editorial Apress.

Referencias

[1] Gizmodo, “Tennis for two”, http://gizmodo.com/5080541/retromodo-tennis-for-two-the-worlds-first-graphical-videogame [Último acceso Julio 2011]

[2] About, “The History of Spacewar”, http://inventors.about.com/od/sstartinventions/a/Spacewar.htm [Último acceso Julio 2011]

[3] Arcade-museum, “Pong”, http://www.arcade-museum.com/game_detail.php?game_id=9074 [Último acceso Julio 2011]

[4] Wikipedia, “Atari Pong”, http://es.wikipedia.org/wiki/Atari_Pong [Último acceso Julio 2011]

[5] Maquina del tiempo, “Historia de Atari”, http://www.puntodepartida.com/retroinformatica/maquinadeltiempo/atari2600.php [Último acceso Julio 2011]

[6] Indice Latino, “Máquinas Recreativas”, http://indicelatino.com/juegos/historia/recreativas/ [Último acceso Julio 2011]

[7] Neoteo, “Nintendo Entertainment System”, http://www.neoteo.com/la-evolucion-de-las-consolas-de-videojuegos [Último acceso Julio 2011]

[8] Publispain, “Historia de Mario Bros”, http://www.publispain.com/mario-bros/historia_de_mario_bros.html [Último acceso Julio 2011]

[9] Entretente, “Top 10 Video Game Consoles of All Time”, http://www.entretente.net/externo/79/Historia-Videoconsolas [Último acceso Julio 2011]

[10] Indice Latino, “Videoconsolas Portatiles”, http://indicelatino.com/juegos/historia/portatiles/ [Último acceso Julio 2011]

[11] Nintendo, “La Historia de Nintendo, 1980”, http://www.nintendo.es/NOE/es_ES/service/la_historia_de_nintendo_9911.html [Último acceso Julio 2011]

[12] Wikipedia, “Game & Watch”, http://es.wikipedia.org/wiki/Game_%26_Watch [Último acceso Julio 2011]

[13] Nintendo, “La Historia de Nintendo, 1989”, http://www.nintendo.es/NOE/es_ES/service/la_historia_de_nintendo_9911.html [Último acceso Julio 2011]

[14] Indice Latino, “Game Boy”, http://indicelatino.com/juegos/historia/portatiles/ [Último acceso Julio 2011]

[15] Indice Latino, “Atari Lynx”, http://indicelatino.com/juegos/historia/portatiles/ [Último acceso Julio 2011]

[16] Retrogames, “PSP”, http://www.retrogames.cl/portatiles.html [Último acceso Julio 2011]

[17] Indice Latino, “N-Gage”, http://indicelatino.com/juegos/historia/portatiles/ [Último acceso Julio 2011]

[18] PocketGamer, “Origen de los juegos en los móviles”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10619 [Último acceso Agosto 2011]

[19] Wikipedia, “Snake”, http://es.wikipedia.org/wiki/Snake_(videojuego) [Último acceso Agosto 2011]

[20] PocketGamer, “Tetris y Snake en los primeros móviles”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10619 [Último acceso Agosto 2011]

[21] Wikipedia, “Wireless Application Protocol”, http://es.wikipedia.org/wiki/Wireless_Application_Protocol [Último acceso Agosto 2011]

[22] Catarina, “Acceso a Internet Inalámbrico”, http://catarina.udlap.mx/u_dl_a/tales/documentos/msp/enriquez_d_c/capitulo4.pdf [Último acceso Agosto 2011]

[23] Wikipedia, “Open Mobile Alliance”, http://es.wikipedia.org/wiki/Open_Mobile_Alliance [Último acceso Agosto 2011]

[24] Wikipedia, “I-mode”, http://es.wikipedia.org/wiki/I-mode [Último acceso Agosto 2011]

[25] Discovery, “Internet”, http://www.tudiscovery.com/internet/mil-cien-millones-de-usuarios.shtml [Último acceso Agosto 2011]

[26] PocketGamer, “Conexión infrarrojos”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10619 [Último acceso Agosto 2011]

[27] PocketGamer, “Empresas sector juegos móviles, 2000”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10626 [Último acceso Agosto 2011]

[28] Gameloft, “Historia de Gameloft”, http://blog.gameloft.com/latam/index.php/2011/08/10/historia-de-gameloft-parte-1/ [Último acceso Agosto 2011]

[29] PocketGamer, “Videojuegos 2000”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10626 [Último acceso Agosto 2011]

[30] PocketGamer, “Empresas sector juegos móviles, 2001”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10691 [Último acceso Agosto 2011]

[31] PocketGamer, “Problema en la industria del videojuego”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10691 [Último acceso Agosto 2011]

[32] PocketGamer, “Videojuegos 2001”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10691 [Último acceso Agosto 2011]

[33] PocketGamer, “Aparición J2ME”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10705 [Último acceso Agosto 2011]

[34] Leo, “Origen y evolución J2ME, Parte 2”, http://leo.ugr.es/J2ME/INTRO/intro_2.htm [Último acceso Agosto 2011]

[35] Leo, “Origen y evolución J2ME, Parte 3”, http://leo.ugr.es/J2ME/INTRO/intro_3.htm [Último acceso Agosto 2011]

[36] PocketGamer, “Moviles con S.O. Java”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10705 [Último acceso Agosto 2011]

[37] Wikipedia, “Binary Runtime for Wireless”, http://en.wikipedia.org/wiki/Binary_Runtime_Environment_for_Wireless [Último acceso Agosto 2011]

[38] Wikipedia, “Mophun”, http://en.wikipedia.org/wiki/Mophun [Último acceso Agosto 2011]

[39] PocketGamer, “Videojuegos 2002”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10705 [Último acceso Agosto 2011]

[40] Taringa, “La Historia del Celular”, http://www.taringa.net/posts/offtopic/785709/La-Historia--del-Celular.html [Último acceso Agosto 2011]

[41] Nokia, “Nokia N-Gage, salida al mercado”, http://press.nokia.com/2003/10/06/let-the-sales-begin-nokia-n-gagetm-game-deck-sales-to-start/ [Último acceso Agosto 2011]

[42] Wikipedia, “Ridge Racer”,http://en.wikipedia.org/wiki/Ridge_Racer_(series) [Último acceso Agosto 2011]

[43] ITespresso, “Extreme Air Snowboarding”, http://www.itespresso.es/extreme-air-snowboarding-llega-al-sony-ericsson-k500i-16292.html [Último acceso Agosto 2011]

[44] PocketGamer, “Absorciones 2004”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10710 [Último acceso Agosto 2011]

[45] PocketGamer, “Videojuegos 2004”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10710 [Último acceso Agosto 2011]

[46] Feel The Byte, “Gizmondo”, http://www.feelthebyte.com/2007/05/gizmondo-una-gran-consola-un-gran-culebron/ [Último acceso Agosto 2011]

[47]Gamepro, “The 10 Worst-Selling Handhelds of All Time”, http://www.gamepro.com/article/features/125749/the-10-worst-selling-handhelds-of-all-time/ [Último acceso Agosto 2011]

[48] Feel The Byte, “Corrupción Gizmondo”, http://www.feelthebyte.com/2007/05/gizmondo-una-gran-consola-un-gran-culebron/ [Último acceso Agosto 2011]

[49] PocketGamer, “Absorciones 2005”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10712 [Último acceso Agosto 2011]

[50] PocketGamer, “Videojuegos 2005”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10712 [Último acceso Agosto 2011]

[51] PocketGamer, “Adquisición Glu Mobile”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10719 [Último acceso Agosto 2011]

[52] FlagSolution, “JUEGOS 3D EN J2ME”, http://www.flagsolutions.net/pdf/JUEGOS%203D%20EN%20J2ME%20CON%20LA%20NUEVA%20API%203D%20JSR.pdf [Último acceso Agosto 2011]

[53] PocketGamer, “Fragmentación del estándar”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10719 [Último acceso Agosto 2011]

[54] PocketGamer, “Videojuegos 2006”, http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10719 [Último acceso Agosto 2011]

[55] Techradar, “N-Gage mark two”, http://www.techradar.com/news/phone-and-communications/mobile-phones/n-gage-mark-two-like-xbox-live-for-mobile--305146 [Último acceso Agosto 2011]

[56] PocketGamer, “2009 - The year in review: N-Gage”, http://www.pocketgamer.co.uk/r/N-Gage/feature.asp?c=17517 [Último acceso Agosto 2011]

[57] Wikipedia, “Historia del iPhone”,http://es.wikipedia.org/wiki/Historia_del_iPhone [Último acceso Agosto 2011]

[58] Wikipedia, “iPhone”,http://es.wikipedia.org/wiki/IPhone [Último acceso Agosto 2011]

[59] Wikipedia, “App Store”,http://es.wikipedia.org/wiki/App_Store [Último acceso Agosto 2011]

[60] Appel, “iPhone App Store Downloads Top 10 Million in First Weekend”, http://www.apple.com/pr/library/2008/07/14iPhone-App-Store-Downloads-Top-10-Million-in-First-Weekend.html [Último acceso Agosto 2011]

[61] SocialBeat, “App Store: 25,000 apps, 800 million downloads”, http://venturebeat.com/2009/03/17/app-store-25000-apps-800-million-downloads/ [Último acceso Agosto 2011]

[62] OJO Internet, “El primer móvil Android”, http://www.ojointernet.com/noticias/el-primer-movil-android-previsto-para-el-23-de-septiembre/ [Último acceso Agosto 2011]

[63] LaFlecha, “Dream”, http://www.laflecha.net/canales/moviles/el-primer-movil-con-el-sistema-operativo-android-de-google-se-llamara-dream [Último acceso Agosto 2011]

[64] Xataka Android, “Android Market sobrepasa las 250.000 aplicaciones”, http://www.xatakandroid.com/mercado/android-market-sobrepasa-las-250000-aplicaciones [Último acceso Agosto 2011]

[65] NielsenWire, “Android Most Popular Operating System in U.S. Among Recent Smartphone Buyers”, http://blog.nielsen.com/nielsenwire/online_mobile/android-most-popular-operating-system-in-u-s-among-recent-smartphone-buyers/ [Último acceso Agosto 2011]

[66] Wikipedia, “Android”,http://es.wikipedia.org/wiki/Android#cite_ref-16 [Último acceso Agosto 2011]

[67] Bloomberg Businessweek, “Google Buys Android for Its Mobile Arsenal”, http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm [Último acceso Agosto 2011]

[68] Open Handset Alliance, “Industry Leaders Announce Open Platform for Mobile Devices”, http://www.openhandsetalliance.com/press_110507.html [Último acceso Agosto 2011]

[69] Wikipedia, “Open Handset Alliance”,http://es.wikipedia.org/wiki/Open_Handset_Alliance [Último acceso Agosto 2011]

[70] SocialBeat, “App Store: 25,000 apps, 800 million downloads”, http://venturebeat.com/2009/03/17/app-store-25000-apps-800-million-downloads/ [Último acceso Agosto 2011]

[71] Apple Insider, “Canalys: iPhone outsold all Windows Mobile phones in Q2 2009”, http://www.appleinsider.com/articles/09/08/21/canalys_iphone_outsold_all_windows_mobile_phones_in_q2_2009.html [Último acceso Agosto 2011]

[72] ComScore, “Mobile Subscriber Market Share, February 2010”, http://www.comscore.com/esl/Press_Events/Press_Releases/2010/4/comScore_Reports_February_2010_U.S._Mobile_Subscriber_Market_Share [Último acceso Agosto 2011]

[73] ComScore, “Mobile Subscriber Market Share, September 2010”, http://www.comscore.com/esl/Press_Events/Press_Releases/2010/11/comScore_Reports_September_2010_U.S._Mobile_Subscriber_Market_Share [Último acceso Agosto 2011]

[74] Cnet, “More signs iPhone under Android attack”, http://news.cnet.com/8301-13579_3-20012418-37.html [Último acceso Agosto 2011]

[75] Cnet, “Android hits top spot in U.S. smartphone market”, http://news.cnet.com/8301-1035_3-20012627-94.html [Último acceso Agosto 2011]

[76] SiliconNews, “550.000 terminales Android activados cada día”, http://www.siliconnews.es/2011/07/15/550-000-terminales-android-activados-cada-dia/ [Último acceso Agosto 2011]

[77] MailxMail, “Aplicaciones del GPS”, http://www.mailxmail.com/curso-introduccion-gps/aplicaciones-gps [Último acceso Octubre 2011]

[78] Jcea, “Precisión del sistema GPS”, http://www.jcea.es/artic/gps-precision.htm [Último acceso Agosto 2011]

[79] Wikipedia, “Navegación marítima”, http://es.wikipedia.org/wiki/Navegaci%C3%B3n_mar%C3%ADtima [Último acceso Octubre 2011]

[80] Alibaba, “GPS para personas mayores”, http://spanish.alibaba.com/product-gs/watch-gps-for-senior-citizen-old-people-kids-child-parkinson-patience-google-map-software-gps-gprs-sms-tracking-465430486.html [Último acceso Octubre 2011]

[81] Enbusca, “Aplicaciones GPS”, http://enbusca.com/seguimiento-de-sus-vastas-aplicaciones-y-dispositivos-gps/ [Último acceso Octubre 2011]

[82] Tragsa, “GPS/GPRS EN FAUNA SILVESTRE”, http://www.tragsa.es/es/lineas-de-actividad/actuaciones-medioambientales/Paginas/sistemas-de-seguimiento-GPS-GPRS-en-fauna-silvestre.aspx [Último acceso Octubre 2011]

[83] http://ocw.upm.es/ingenieria-cartografica-geodesica-y-fotogrametria/topografia-ii/Teoria_GPS_Tema_12.pdf [Último acceso Octubre 2011]

[84] Wikipedia, “Sistema de posicionamiento global”, http://es.wikipedia.org/wiki/Sistema_de_posicionamiento_global [Último acceso Agosto 2011]

[85] UPM, “*Aplicaciones Topográficas del G.P.S.*”, http://www.galsys.co.uk/es/time-reference/gps-and-ntp/gps-for-computer-timing.html [Último acceso Octubre 2011]

[86] Galleon Systems, “GPS para aplicaciones de sincronización PC”, http://www.nosolounix.com/2011/04/gps-para-android.html [Último acceso Octubre 2011]

[87] TuAndroid, “GPS Essentials”, http://www.tuandroid.com/gps-essentials-para-android/ [Último acceso Septiembre 2011]

[88] Android Market, “Car Locator”, https://market.android.com/details?id=com.edwardkim.android.carlocatorfull&hl=es [Último acceso Septiembre 2011]

[89] Indet, “Indet Aplication”, https://www.indetweb.com [Último acceso Septiembre 2011]

[90] Android.es, “Kulturize”, http://www.android.es/kulturize-todos-los-eventos-culturales-de-tu-zona-en-una-unica-agenda.html#axzz1ZnklsdLV [Último acceso Septiembre 2011]

[91] Layar, “Layar Aplication”, http://www.layar.com [Último acceso Septiembre 2011]

[92] Microsiervos, “Layar”, http://www.microsiervos.com/archivo/tecnologia/laya-realidad-aumentada-android.html [Último acceso Septiembre 2011]

[93] Xakata Móvil, “Layar”, http://www.xatakamovil.com/aplicaciones/layar-primer-navegador-android-con-realidad-aumentada [Último acceso Septiembre 2011]

[94] Societic, “Twinttaround”, http://www.societic.com/2011/04/twittaround-como-mezclar-twitter-y-la-realidad-aumentada-en-una-misma-aplicacion/ [Último acceso Septiembre 2011]

[95] Xakata Móvil, “Twinttaround”,http://www.xatakamovil.com/aplicaciones/twittaround-twitter-y-realidad-aumentada-en-un-iphone-3gs [Último acceso Septiembre 2011]

[96] America Learning & Media, “Wikitude World Browser”, http://www.americalearningmedia.com/component/content/article/69-tester/264-13-aplicaciones-de-realidad-aumentada [Último acceso Septiembre 2011]

[97] Wikitude, “Worlds”, http://www.wikitude.com/worlds [Último acceso Septiembre 2011]

[98] Wikitude, “Wikitude MyWorld”, http://www.wikitude.com/enwikitude-evernote-trunk-conference-august-18-san-francisco [Último acceso Septiembre 2011]

[99] Wikitude, “Wikitude World Browser”, http://www.wikitude.com/wikitude-world-browser-augmented-reality [Último acceso Septiembre 2011]

[100] Wearable Computer Lab, “ARQuake: Interactive Outdoor Augmented Reality Collaboration System”, http://wearables.unisa.edu.au/projects/arquake/ [Último acceso Septiembre 2011]

[101] Blast Theory, “CAN YOU SEE ME NOW?”, http://www.blasttheory.co.uk/bt/work_cysmn.html [Último acceso Septiembre 2011]

[102] Wearable Computer Lab, “What kind of tracking equipment do you use for this? GPS is only accurate to 5 metres!”, http://wearables.unisa.edu.au/projects/arquake/ [Último acceso Septiembre 2011]

[103] Augmented Planet, “Augmented Reality Golf”, http://www.augmentedplanet.com/2010/05/augmented-reality-golf-but-not-what-you-think/ [Último acceso Octubre 2011]

[104] Gigantic Mechanic, “Gigaputt”, http://www.giganticmechanic.com/games_gigaputt.html [Último acceso Octubre 2011]

[105] Spectrekking, “SpecTrek”, http://www.spectrekking.com/ [Último acceso Octubre 2011]

[106] MilaWeb, “Zombies, Run!”, http://www.milaweb.com/blog/jugar-juegos-zombies-android/ [Último acceso Octubre 2011]

[107] Geek, “Zombies, Run!”, http://www.geek.com/articles/mobile/zombies-run-iphone-app-will-keep-you-running-literally-20110914/ [Último acceso Octubre 2011]

[108] Zombies Run!, “Zombies, Run! Game”, http://www.zombiesrungame.com/ [Último acceso Octubre 2011]

Anexos

Anexo A. Planificación

La planificación de un proyecto es fundamental para el buen desarrollo de éste. Cumplir con los plazos previstos, saber qué partes son críticas si no se terminan a tiempo y a qué puede afectar, garantizará el resultado exitoso del mismo.

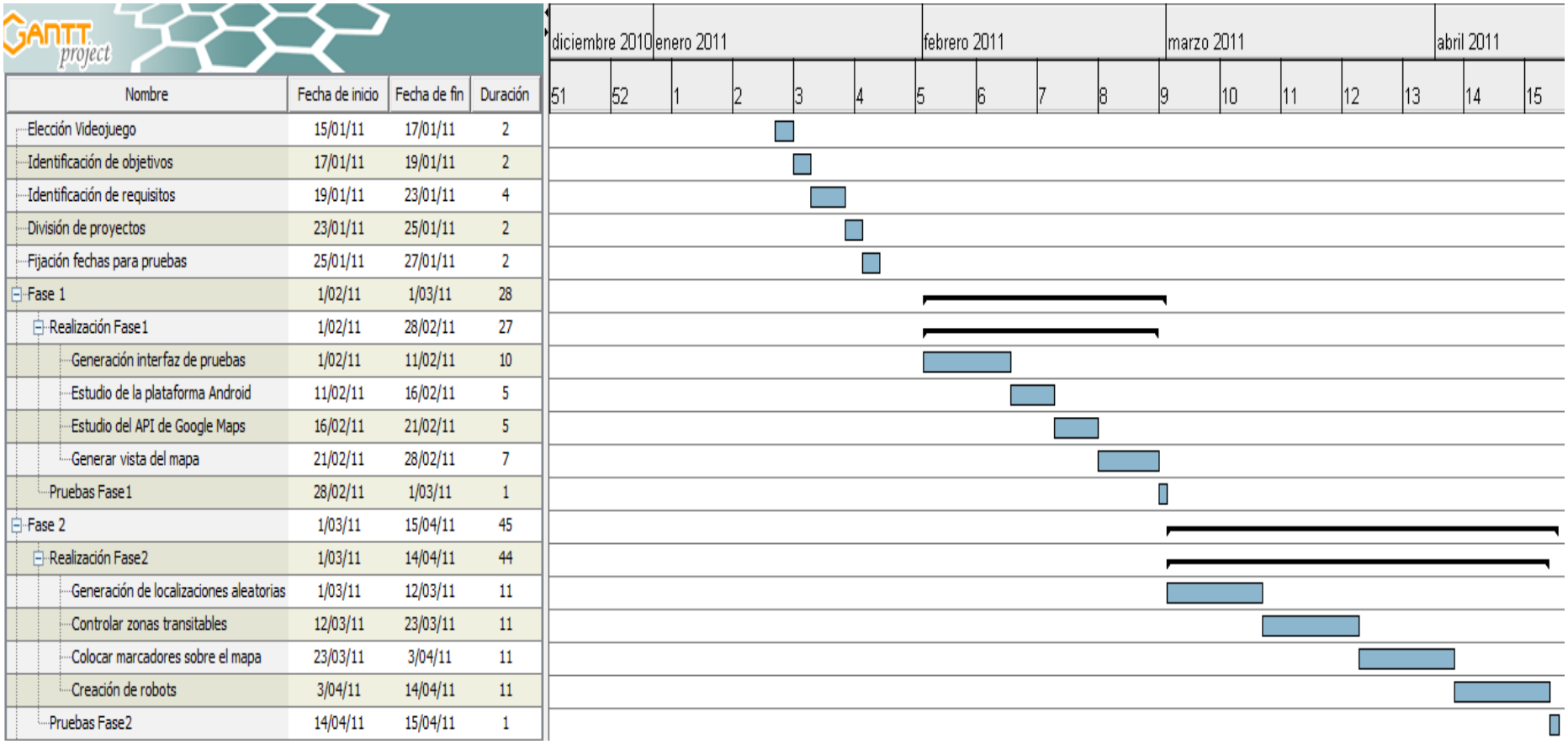
A continuación se presentan las planificaciones iniciales y finales, para comprobar si se han cumplido los plazos previstos y, si no es así qué, partes han sido las causantes. Diferenciaremos entre la planificación del videojuego completo y la planificación de este proyecto.

A.1. Planificación inicial

Planificación inicial del videojuego:

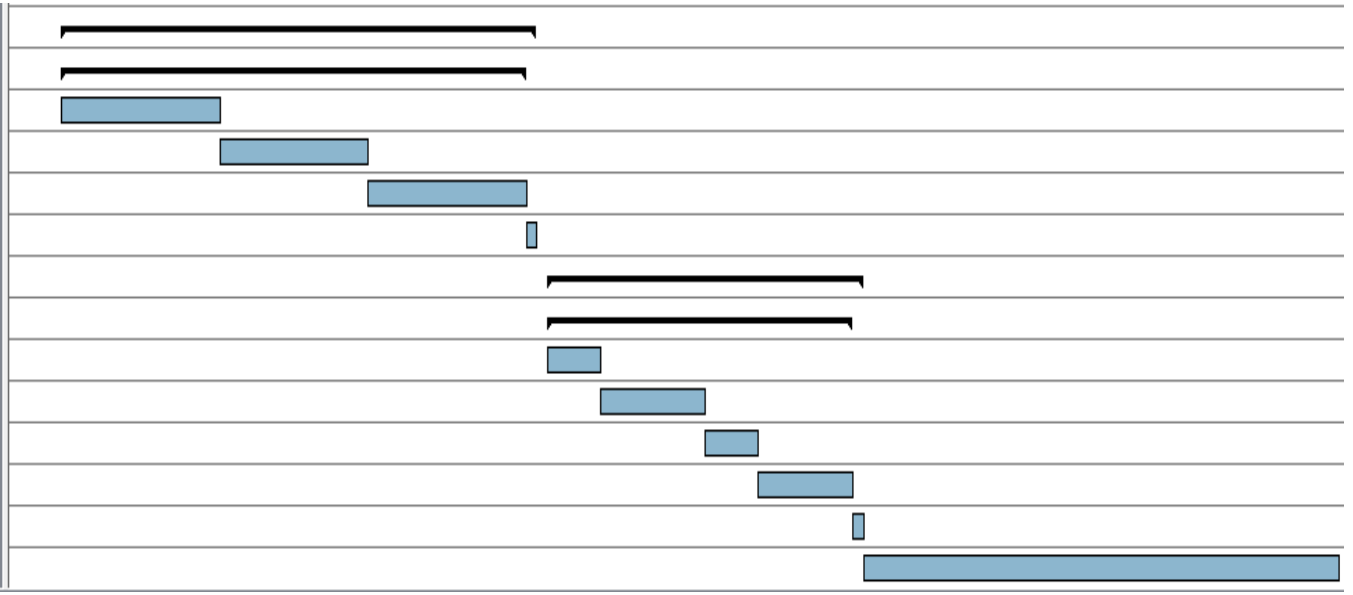
Nombre	Fecha de inicio	Fecha de fin	Duración
Elección Videojuego	15/01/11	17/01/11	2
Identificación de objetivos	17/01/11	19/01/11	2
Identificación de requisitos	19/01/11	23/01/11	4
División de proyectos	23/01/11	25/01/11	2
Fijación fechas para pruebas	25/01/11	27/01/11	2
Fase 1	1/02/11	1/03/11	28
Realización Fase1	1/02/11	28/02/11	27
Pruebas Fase1	28/02/11	1/03/11	1
Fase 2	1/03/11	15/04/11	45
Realización Fase2	1/03/11	14/04/11	44
Pruebas Fase2	14/04/11	15/04/11	1
Fase 3	16/04/11	31/05/11	45
Realización Fase3	16/04/11	30/05/11	44
Pruebas Fase3	30/05/11	31/05/11	1
Fase4	1/06/11	1/07/11	30
Realización Fase4	1/06/11	30/06/11	29
Pruebas Fase4	30/06/11	1/07/11	1
Memoria	1/07/11	15/08/11	45
Pruebas Finales	15/08/11	30/08/11	15

Antes de comenzar con la realización de ambos proyectos por separado, hay una parte común. Seguidamente se divide el videojuego en cuatro fases, en cada una de las cuales se desarrollarán los proyectos por separado, y al final de cada una se realizarán las pruebas oportunas para la comprobación del correcto funcionamiento del videojuego completo.



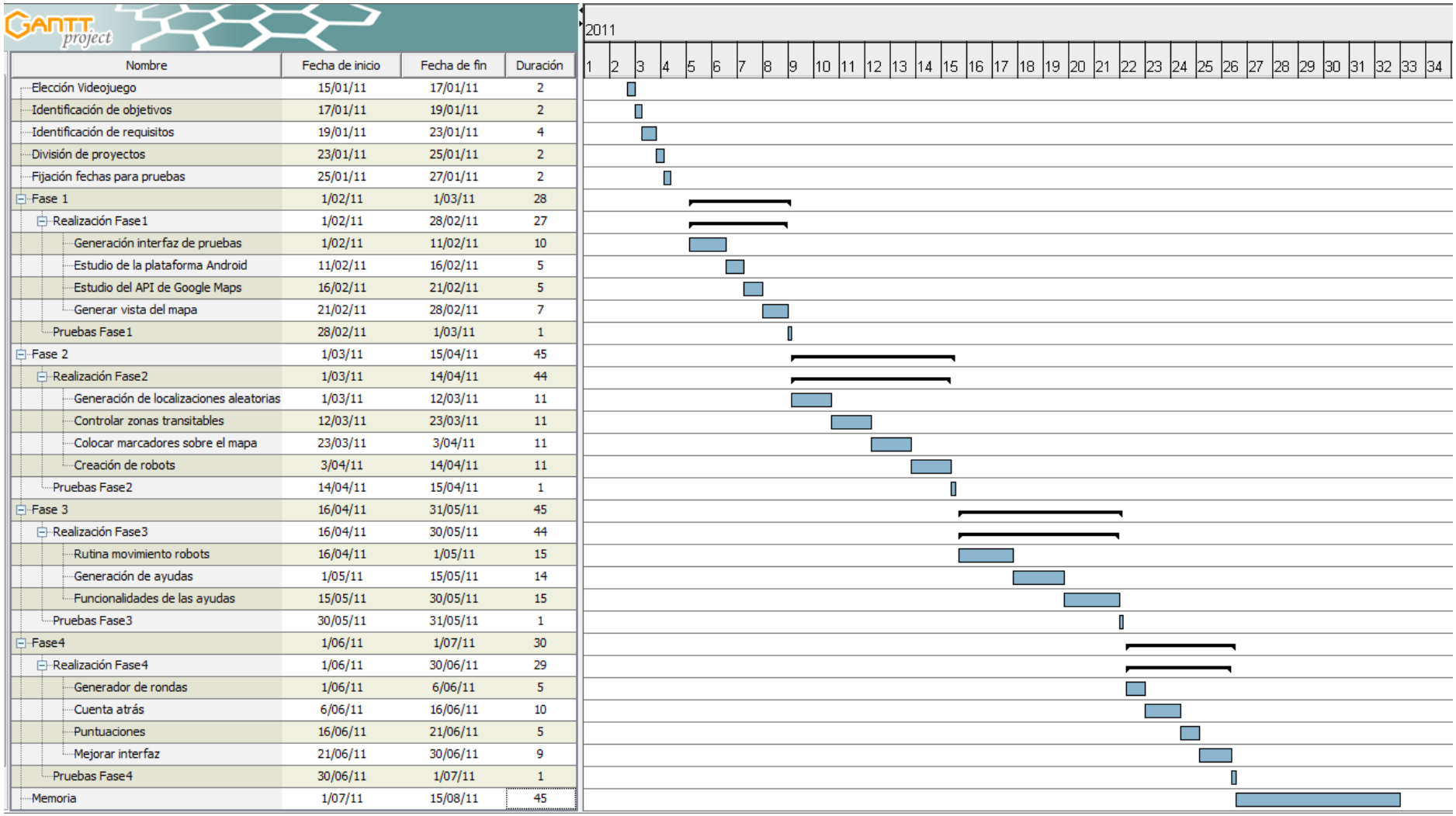
Nombre	Fecha de inicio	Fecha de fin	Duración
Elección Videojuego	15/01/11	17/01/11	2
Identificación de objetivos	17/01/11	19/01/11	2
Identificación de requisitos	19/01/11	23/01/11	4
División de proyectos	23/01/11	25/01/11	2
Fijación fechas para pruebas	25/01/11	27/01/11	2
Fase 1	1/02/11	1/03/11	28
Realización Fase1	1/02/11	28/02/11	27
Generación interfaz de pruebas	1/02/11	11/02/11	10
Estudio de la plataforma Android	11/02/11	16/02/11	5
Estudio del API de Google Maps	16/02/11	21/02/11	5
Generar vista del mapa	21/02/11	28/02/11	7
Pruebas Fase1	28/02/11	1/03/11	1
Fase 2	1/03/11	15/04/11	45
Realización Fase2	1/03/11	14/04/11	44
Generación de localizaciones aleatorias	1/03/11	12/03/11	11
Controlar zonas transitables	12/03/11	23/03/11	11
Colocar marcadores sobre el mapa	23/03/11	3/04/11	11
Creación de robots	3/04/11	14/04/11	11
Pruebas Fase2	14/04/11	15/04/11	1

Fase 3	16/04/11	31/05/11	45
Realización Fase3	16/04/11	30/05/11	44
Rutina movimiento robots	16/04/11	1/05/11	15
Generación de ayudas	1/05/11	15/05/11	14
Funcionalidades de las ayudas	15/05/11	30/05/11	15
Pruebas Fase3	30/05/11	31/05/11	1
Fase4	1/06/11	1/07/11	30
Realización Fase4	1/06/11	30/06/11	29
Generador de rondas	1/06/11	6/06/11	5
Cuenta atrás	6/06/11	16/06/11	10
Puntuaciones	16/06/11	21/06/11	5
Mejorar interfaz	21/06/11	30/06/11	9
Pruebas Fase4	30/06/11	1/07/11	1
Memoria	1/07/11	15/08/11	45



A continuación se presenta la planificación inicial del proyecto, especificando las tareas a realizar en cada una de las fases.

Nombre	Fecha de inicio	Fecha de fin	Duración
Elección Videojuego	15/01/11	17/01/11	2
Identificación de objetivos	17/01/11	19/01/11	2
Identificación de requisitos	19/01/11	23/01/11	4
División de proyectos	23/01/11	25/01/11	2
Fijación fechas para pruebas	25/01/11	27/01/11	2
Fase 1	1/02/11	1/03/11	28
Realización Fase1	1/02/11	28/02/11	27
Generación interfaz de pruebas	1/02/11	11/02/11	10
Estudio de la plataforma Android	11/02/11	16/02/11	5
Estudio del API de Google Maps	16/02/11	21/02/11	5
Generar vista del mapa	21/02/11	28/02/11	7
Pruebas Fase1	28/02/11	1/03/11	1
Fase 2	1/03/11	15/04/11	45
Realización Fase2	1/03/11	14/04/11	44
Generación de localizaciones aleatorias	1/03/11	12/03/11	11
Controlar zonas transitables	12/03/11	23/03/11	11
Colocar marcadores sobre el mapa	23/03/11	3/04/11	11
Creación de robots	3/04/11	14/04/11	11
Pruebas Fase2	14/04/11	15/04/11	1
Fase 3	16/04/11	31/05/11	45
Realización Fase3	16/04/11	30/05/11	44
Rutina movimiento robots	16/04/11	1/05/11	15
Generación de ayudas	1/05/11	15/05/11	14
Funcionalidades de las ayudas	15/05/11	30/05/11	15
Pruebas Fase3	30/05/11	31/05/11	1
Fase4	1/06/11	1/07/11	30
Realización Fase4	1/06/11	30/06/11	29
Generador de rondas	1/06/11	6/06/11	5
Cuenta atrás	6/06/11	16/06/11	10
Puntuaciones	16/06/11	21/06/11	5
Mejorar interfaz	21/06/11	30/06/11	9
Pruebas Fase4	30/06/11	1/07/11	1
Memoria	1/07/11	15/08/11	45



Nombre	Fecha de inicio	Fecha de fin	Duración
Elección Videojuego	15/01/11	17/01/11	2
Identificación de objetivos	17/01/11	19/01/11	2
Identificación de requisitos	19/01/11	23/01/11	4
División de proyectos	23/01/11	25/01/11	2
Fijación fechas para pruebas	25/01/11	27/01/11	2
Fase 1	1/02/11	1/03/11	28
Realización Fase1	1/02/11	28/02/11	27
Generación interfaz de pruebas	1/02/11	11/02/11	10
Estudio de la plataforma Android	11/02/11	16/02/11	5
Estudio del API de Google Maps	16/02/11	21/02/11	5
Generar vista del mapa	21/02/11	28/02/11	7
Pruebas Fase1	28/02/11	1/03/11	1
Fase 2	1/03/11	15/04/11	45
Realización Fase2	1/03/11	14/04/11	44
Generación de localizaciones aleatorias	1/03/11	12/03/11	11
Controlar zonas transitables	12/03/11	23/03/11	11
Colocar marcadores sobre el mapa	23/03/11	3/04/11	11
Creación de robots	3/04/11	14/04/11	11
Pruebas Fase2	14/04/11	15/04/11	1
Fase 3	16/04/11	31/05/11	45
Realización Fase3	16/04/11	30/05/11	44
Rutina movimiento robots	16/04/11	1/05/11	15
Generación de ayudas	1/05/11	15/05/11	14
Funcionalidades de las ayudas	15/05/11	30/05/11	15
Pruebas Fase3	30/05/11	31/05/11	1
Fase4	1/06/11	1/07/11	30
Realización Fase4	1/06/11	30/06/11	29
Generador de rondas	1/06/11	6/06/11	5
Cuenta atrás	6/06/11	16/06/11	10
Puntuaciones	16/06/11	21/06/11	5
Mejorar interfaz	21/06/11	30/06/11	9
Pruebas Fase4	30/06/11	1/07/11	1
Memoria	1/07/11	15/08/11	45

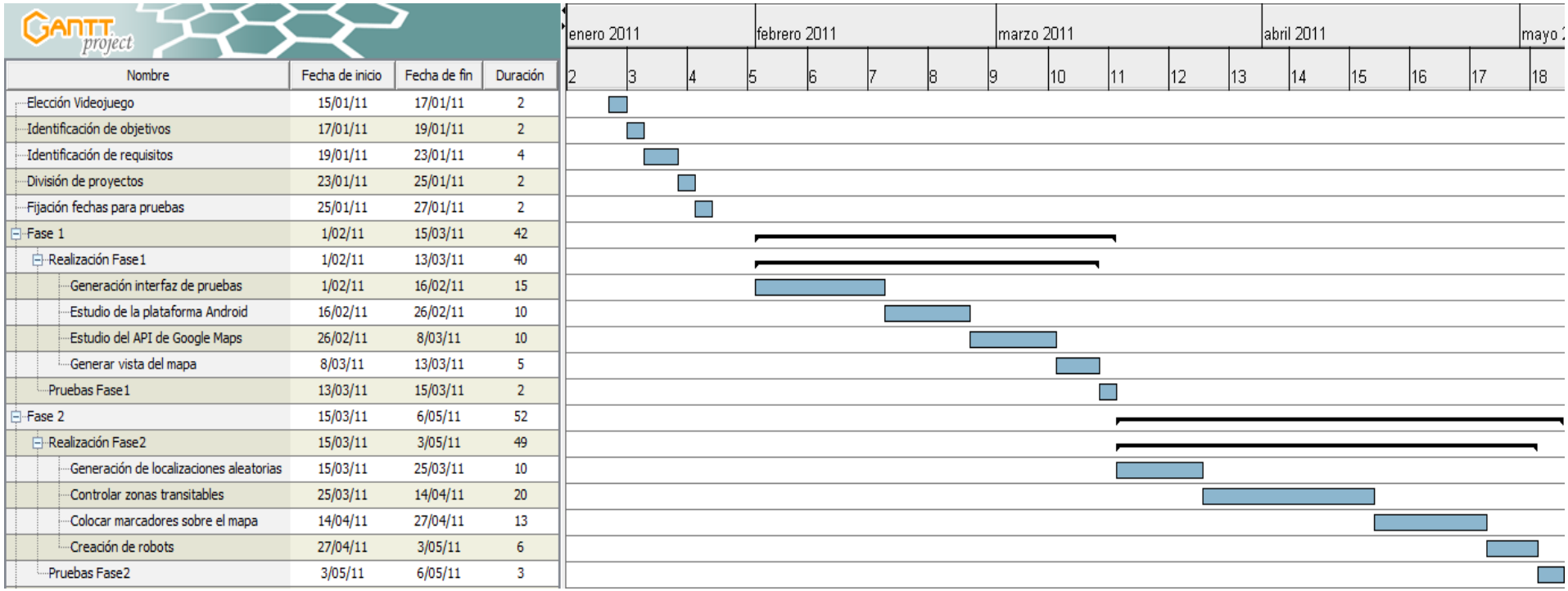
A2. Planificación final

A continuación se muestra la Planificación final de igual manera que se expuso la inicial, con las alteraciones de tiempo que se han producido retrasando la entrega final del proyecto.

Primero se expone de nuevo el listado de tareas, ya que se han añadido algún hito.

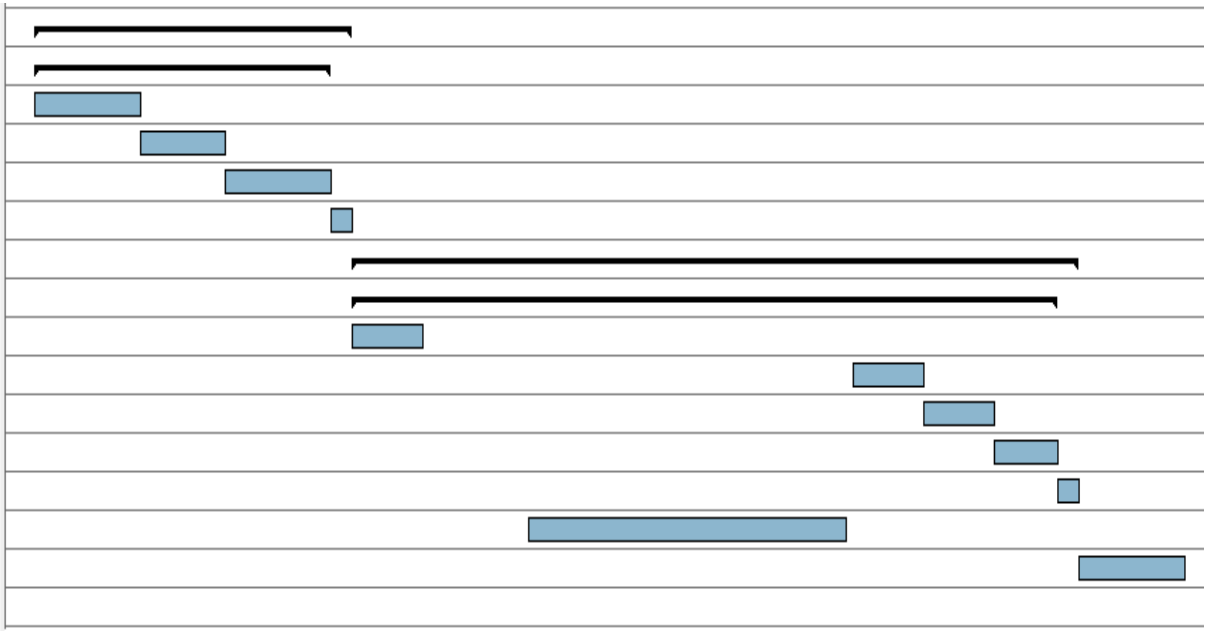
Nombre	Fecha de inicio	Fecha de fin	Duración
Elección Videojuego	15/01/11	17/01/11	2
Identificación de objetivos	17/01/11	19/01/11	2
Identificación de requisitos	19/01/11	23/01/11	4
División de proyectos	23/01/11	25/01/11	2
Fijación fechas para pruebas	25/01/11	27/01/11	2
Fase 1	1/02/11	15/03/11	42
Realización Fase1	1/02/11	13/03/11	40
Pruebas Fase1	13/03/11	15/03/11	2
Fase 2	15/03/11	6/05/11	52
Realización Fase2	15/03/11	3/05/11	49
Pruebas Fase2	3/05/11	6/05/11	3
Fase 3	6/05/11	20/06/11	45
Realización Fase3	6/05/11	17/06/11	42
Pruebas Fase3	17/06/11	20/06/11	3
Fase4	20/06/11	1/10/11	103
Realización Fase4	20/06/11	28/09/11	100
Pruebas Fase4	28/09/11	1/10/11	3
Memoria	15/07/11	29/08/11	45
Modificación Memoria	1/10/11	16/10/11	15
Pruebas Finales	15/11/11	25/11/11	10

En el siguiente gráfico se muestran las diferentes fases con los hitos que constituyen cada una de ellas.



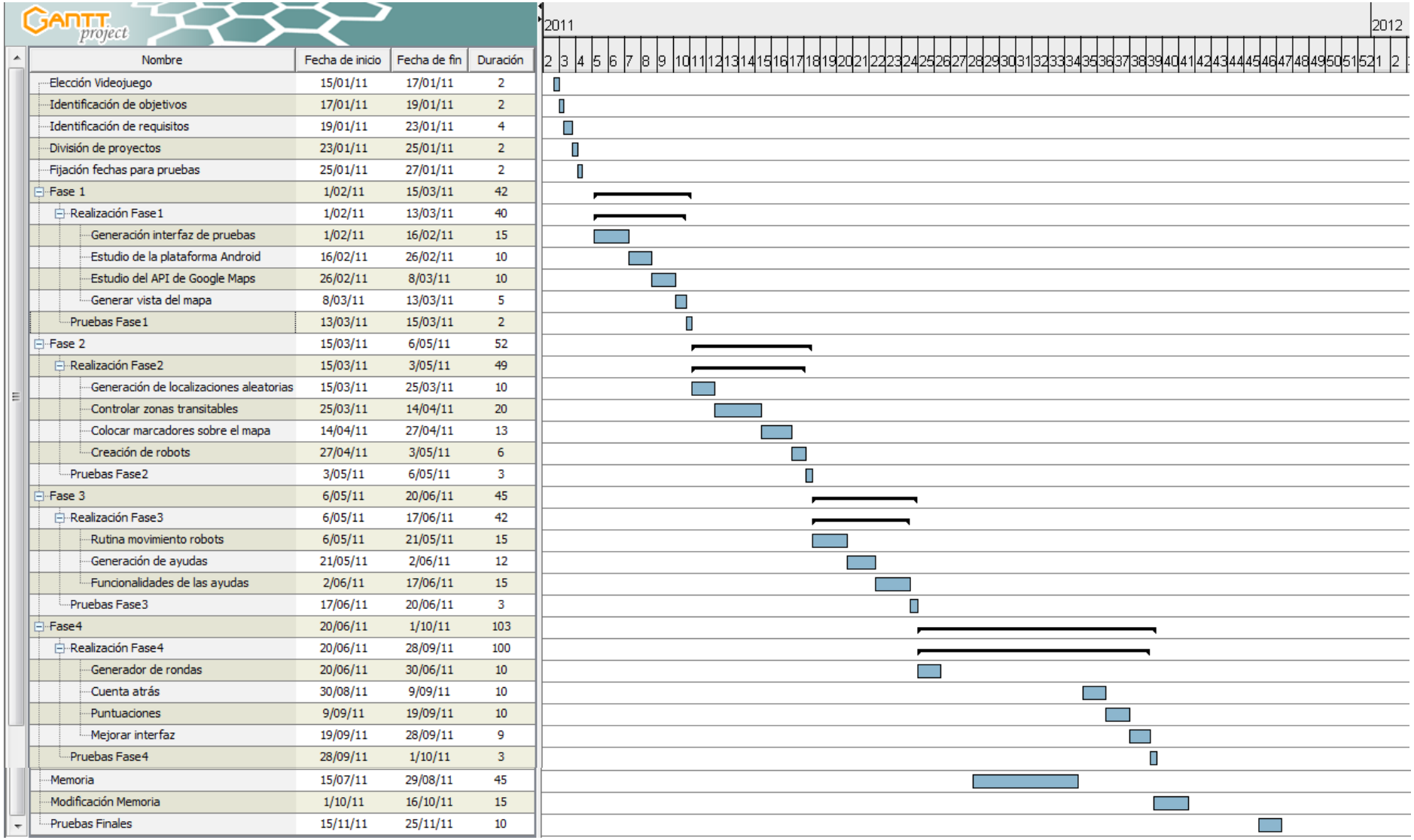
Nombre	Fecha de inicio	Fecha de fin	Duración
Elección Videojuego	15/01/11	17/01/11	2
Identificación de objetivos	17/01/11	19/01/11	2
Identificación de requisitos	19/01/11	23/01/11	4
División de proyectos	23/01/11	25/01/11	2
Fijación fechas para pruebas	25/01/11	27/01/11	2
Fase 1	1/02/11	15/03/11	42
Realización Fase1	1/02/11	13/03/11	40
Generación interfaz de pruebas	1/02/11	16/02/11	15
Estudio de la plataforma Android	16/02/11	26/02/11	10
Estudio del API de Google Maps	26/02/11	8/03/11	10
Generar vista del mapa	8/03/11	13/03/11	5
Pruebas Fase1	13/03/11	15/03/11	2
Fase 2	15/03/11	6/05/11	52
Realización Fase2	15/03/11	3/05/11	49
Generación de localizaciones aleatorias	15/03/11	25/03/11	10
Controlar zonas transitables	25/03/11	14/04/11	20
Colocar marcadores sobre el mapa	14/04/11	27/04/11	13
Creación de robots	27/04/11	3/05/11	6
Pruebas Fase2	3/05/11	6/05/11	3

Fase 3	6/05/11	20/06/11	45
Realización Fase3	6/05/11	17/06/11	42
Rutina movimiento robots	6/05/11	21/05/11	15
Generación de ayudas	21/05/11	2/06/11	12
Funcionalidades de las ayudas	2/06/11	17/06/11	15
Pruebas Fase3	17/06/11	20/06/11	3
Fase4	20/06/11	1/10/11	103
Realización Fase4	20/06/11	28/09/11	100
Generador de rondas	20/06/11	30/06/11	10
Cuenta atrás	30/08/11	9/09/11	10
Puntuaciones	9/09/11	19/09/11	10
Mejorar interfaz	19/09/11	28/09/11	9
Pruebas Fase4	28/09/11	1/10/11	3
Memoria	15/07/11	29/08/11	45
Modificación Memoria	1/10/11	16/10/11	15
Pruebas Finales	15/11/11	25/11/11	10



Por último, se presenta la planificación final del proyecto, especificando las tareas a realizar en cada una de las fases.

Nombre	Fecha de inicio	Fecha de fin	Duración
Elección Videojuego	15/01/11	17/01/11	2
Identificación de objetivos	17/01/11	19/01/11	2
Identificación de requisitos	19/01/11	23/01/11	4
División de proyectos	23/01/11	25/01/11	2
Fijación fechas para pruebas	25/01/11	27/01/11	2
Fase 1	1/02/11	15/03/11	42
Realización Fase1	1/02/11	13/03/11	40
Generación interfaz de pruebas	1/02/11	16/02/11	15
Estudio de la plataforma Android	16/02/11	26/02/11	10
Estudio del API de Google Maps	26/02/11	8/03/11	10
Generar vista del mapa	8/03/11	13/03/11	5
Pruebas Fase1	13/03/11	15/03/11	2
Fase 2	15/03/11	6/05/11	52
Realización Fase2	15/03/11	3/05/11	49
Generación de localizaciones aleatorias	15/03/11	25/03/11	10
Controlar zonas transitables	25/03/11	14/04/11	20
Colocar marcadores sobre el mapa	14/04/11	27/04/11	13
Creación de robots	27/04/11	3/05/11	6
Pruebas Fase2	3/05/11	6/05/11	3
Fase 3	6/05/11	20/06/11	45
Realización Fase3	6/05/11	17/06/11	42
Rutina movimiento robots	6/05/11	21/05/11	15
Generación de ayudas	21/05/11	2/06/11	12
Funcionalidades de las ayudas	2/06/11	17/06/11	15
Pruebas Fase3	17/06/11	20/06/11	3
Fase4	20/06/11	1/10/11	103
Realización Fase4	20/06/11	28/09/11	100
Generador de rondas	20/06/11	30/06/11	10
Cuenta atrás	30/08/11	9/09/11	10
Puntuaciones	9/09/11	19/09/11	10
Mejorar interfaz	19/09/11	28/09/11	9
Pruebas Fase4	28/09/11	1/10/11	3
Memoria	15/07/11	29/08/11	45
Modificación Memoria	1/10/11	16/10/11	15
Pruebas Finales	15/11/11	25/11/11	10



Nombre	Fecha de inicio	Fecha de fin	Duración
Elección Videojuego	15/01/11	17/01/11	2
Identificación de objetivos	17/01/11	19/01/11	2
Identificación de requisitos	19/01/11	23/01/11	4
División de proyectos	23/01/11	25/01/11	2
Fijación fechas para pruebas	25/01/11	27/01/11	2
Fase 1	1/02/11	15/03/11	42
Realización Fase1	1/02/11	13/03/11	40
Generación interfaz de pruebas	1/02/11	16/02/11	15
Estudio de la plataforma Android	16/02/11	26/02/11	10
Estudio del API de Google Maps	26/02/11	8/03/11	10
Generar vista del mapa	8/03/11	13/03/11	5
Pruebas Fase1	13/03/11	15/03/11	2
Fase 2	15/03/11	6/05/11	52
Realización Fase2	15/03/11	3/05/11	49
Generación de localizaciones aleatorias	15/03/11	25/03/11	10
Controlar zonas transitables	25/03/11	14/04/11	20
Colocar marcadores sobre el mapa	14/04/11	27/04/11	13
Creación de robots	27/04/11	3/05/11	6
Pruebas Fase2	3/05/11	6/05/11	3
Fase 3	6/05/11	20/06/11	45
Realización Fase3	6/05/11	17/06/11	42
Rutina movimiento robots	6/05/11	21/05/11	15
Generación de ayudas	21/05/11	2/06/11	12
Funcionalidades de las ayudas	2/06/11	17/06/11	15
Pruebas Fase3	17/06/11	20/06/11	3
Fase4	20/06/11	1/10/11	103
Realización Fase4	20/06/11	28/09/11	100
Generador de rondas	20/06/11	30/06/11	10
Cuenta atrás	30/08/11	9/09/11	10
Puntuaciones	9/09/11	19/09/11	10
Mejorar interfaz	19/09/11	28/09/11	9
Pruebas Fase4	28/09/11	1/10/11	3
Memoria	15/07/11	29/08/11	45
Modificación Memoria	1/10/11	16/10/11	15
Pruebas Finales	15/11/11	25/11/11	10

Anexo B. Presupuesto

Atendiendo a la planificación final se puede calcular a cuánto asciende el presupuesto del videojuego. Se calculará el presupuesto del videojuego completo, para posteriormente poder hacer un estudio de los posibles beneficios que se pueden obtener.

La duración del desarrollo completo del videojuego Invasión Androide asciende a un total de 324 días. 45 días corresponden al trabajo realizado por un analista y los 279 restantes al trabajo de programadores. Se puede aproximar que cada día el analista haya dedicado 5 horas de trabajo, y los programadores 3 y media. Por su parte el jefe de proyecto ha dedicado al sistema entero un total de 25 horas.

Cargo Profesional	Retribución (€/año)	Retribución (€/h)	Horas trabajadas	Total (€)
Jefe de proyecto	40.000	22,2	25	555
Analista	30.000	16,6	225	3.735
Programadores	20.000	11,2	976	10.931,2

Lo que asciende a un total de 26.342,4€ para la elaboración del videojuego.

Se debe hacer un estudio de los posibles beneficios que se pueden obtener con la elaboración del videojuego. Al tratarse de un videojuego Android, la manera de ofrecerlo al público es a través de la aplicación Android Market. Para obtener beneficios habría que subir la aplicación con un precio por descarga. Teniendo en cuenta que los precios de este tipo de videojuegos, siempre que no son gratuitos, son bastante bajos, un precio no descabellado podría ser de 60 céntimos. Para amortizar el coste, habría que tener como mínimo 43.904 descargas, lo que supondría en un año a una media de 3.659 descargas mensuales.

Sin embargo hay que apuntar, que como bien se expuso en el capítulo 2 de esta memoria, hay varias aplicaciones de realidad aumentada con características similares a este proyecto que aparecen de forma gratuita para descargar desde el Android Market. Por tanto, es de esperar, que no se produzcan las descargas esperadas y que se tarde más de un año en amortizar el videojuego, si es que esto llega a producirse.

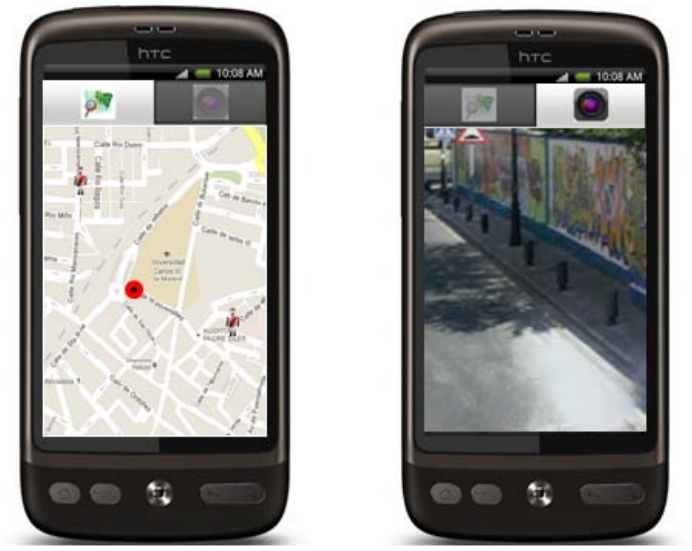
Una opción más viable para sacar beneficios del videojuego, sería buscar una empresa que quisiera publicitarse en él, que cubriera con los gastos que ha supuesto su elaboración más una cantidad establecida por descarga. De esta manera, se podría establecer la descarga del videojuego gratuitamente y con ello garantizar que el número de descargas seguramente será mucho mayor que fijándole un precio.

Anexo C. Manual de Usuario

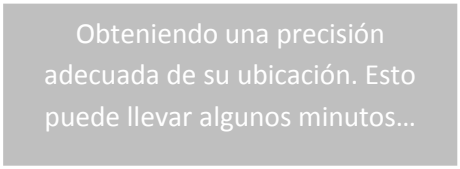
Al iniciar el videojuego Invasión Robótica, se abre la pantalla principal, con las opciones que aparecen en la figura:

El botón instrucciones, muestra las instrucciones del juego, Opciones audio, la posibilidad de activar o desactivar los efectos de sonido, Records, muestra las 3 puntuaciones más altas conseguidas, Créditos, los créditos correspondientes al juego, y mediante el botón salir se cierra la aplicación.

Mediante el botón Jugar, se inicia una nueva partida. Se abre una pantalla con dos pestañas, para poder elegir entre mostrar el mapa con la ubicación del usuario y los robots a capturar, o mostrar la imagen de la cámara para buscar a éstos y capturarlos. Por defecto a parece la imagen del mapa.



Al comenzar una nueva partida, el GPS debe obtener una ubicación del usuario con buena precisión, por lo que se debe esperar el tiempo suficiente hasta conseguirlo. Se mostrará el siguiente mensaje:



Si la señal del GPS se pierde, la aplicación se cierra volviendo al menú principal y avisando al usuario de lo ocurrido:

Los robots y las ayudas serán visualizados cuando se encuentren a una distancia máxima de 20 metros. Sin embargo no podrán ser capturados hasta que no se encuentren una distancia inferior a 7 metros. Cuando esto ocurra, se le mostrará un mensaje al usuario informando de que puede capturar:

Cuando se captura un robot, se activa un efecto sonoro (si los efectos de sonido han sido activados, o si no han sido desactivados), y un mensaje informativo:

De la misma manera, cuando se consigue una ayuda, se activa el efecto sonoro si es preciso y se muestra un mensaje informativo:

Hay 3 tipos de ayuda diferentes, que proporcionan una característica distinta cada una. Pueden diferenciarse por su dibujo, como se muestra en la siguiente figura:

AYUDA	DESCRIPCIÓN
	La puntuación que el usuario consiga durante el minuto y medio seguido a la captura del ítem, se multiplicará por 2
	Los robots permanecen inmóviles durante 45 segundos
	El tiempo deja de decrementarse durante 45 segundos

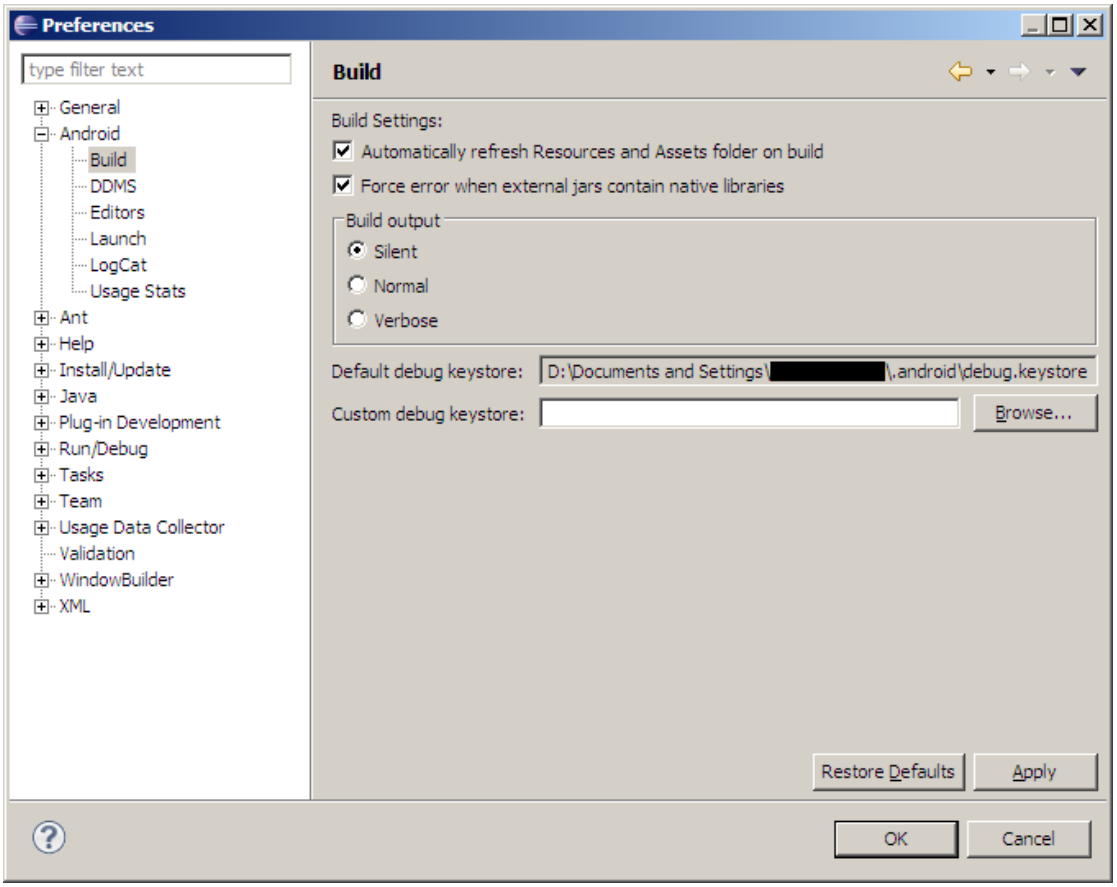
Cada vez que el usuario captura todos los robots pertenecientes a una misma ronda, esta incrementa:

Cuando el tiempo finaliza, se muestra al usuario la puntuación obtenida.

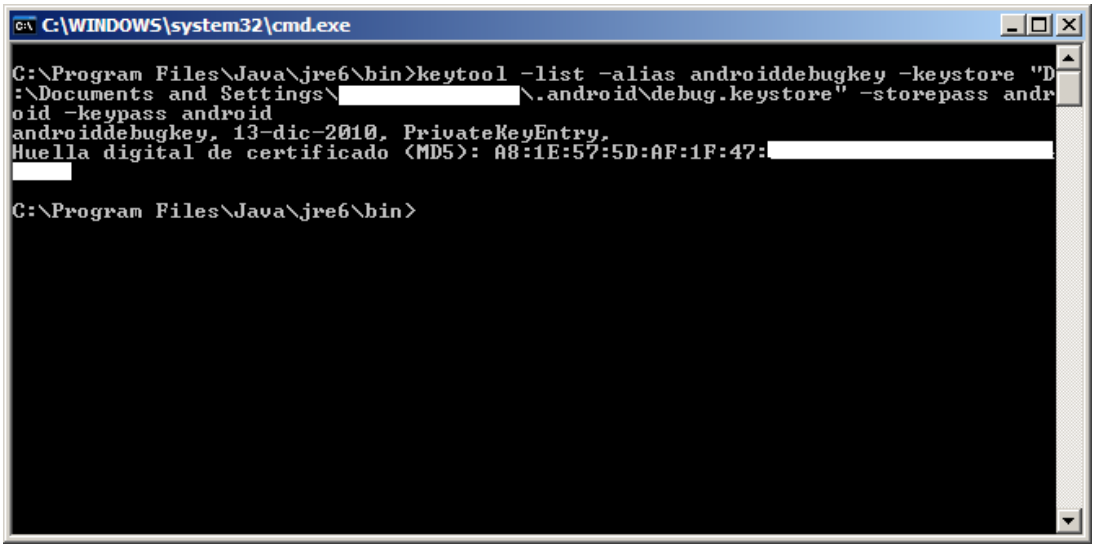
Anexo D. Manual del desarrollador

Obtención API Key

En primer lugar se debe localizar el fichero donde se almacenan los datos del certificado de depuración, llamado debug.keystore. Se puede saber la ruta de este fichero accediendo a las preferencias de Eclipse, sección *Android*, apartado *Build* (mostrado en la siguiente imagen), y copiar la ruta que aparece en el campo "*Default Debug Keystore*":



Una vez se conoce la localización del fichero debug.keystore, se accederá a él mediante la herramienta keytool.exe de java para obtener el *hash* MD5 del certificado. Esto se hará desde la línea de comandos de Windows mediante el siguiente comando que se muestra en la imagen:



Se copiará el dato que aparece identificado como "*Huella digital de certificado (MD5)*" y con éste se accederá a la web (http://code.google.com/android/maps-api-signup.html) de Google para solicitar una clave de uso de la API de Google Maps para depurar las aplicaciones que se desarrollen. Dicha web solicitará la marca MD5 de nuestro certificado y proporcionará la clave de uso de la API.